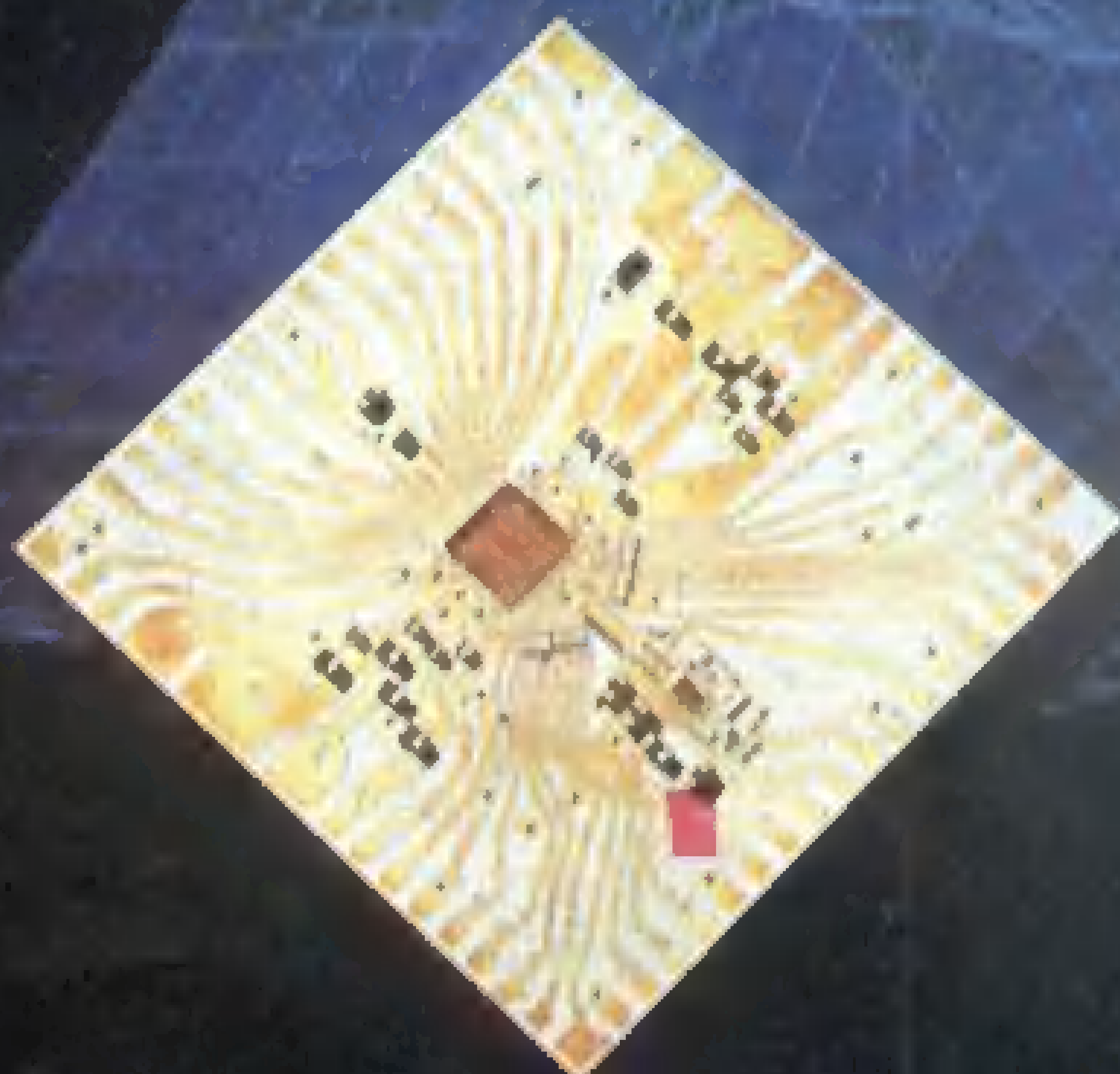
młody TECHNIK

InforMik

MAGAZYN KOMPUTEROWY „MŁODEGO TECHNIKA"

11
1988



Nasz komentarz:

POD OSTRZAŁEM

Zasiadam do pisania niniejszego komentarza bezpośrednio po wyjątkowo budującym przeżyciu, jakim było obejrzenie w wieczornym programie rodzimej telewizji reklam sprzętu komputerowego. Twierdzę bowiem, że z owych reklam można dowiedzieć się o wiele więcej, niż tylko kto co ma do zbycia.

Zgrabna aranżacja i zręczna realizacja nie pozostawiają wątpliwości, że reklamy te są dziełem fachowców od promocji i marketingu. Wiadomo nie od dziś, że dobra reklama opiera się na rozpoznaniu rynku, a więc m.in. także na trafnie wybranym, psychologicznym modelu kandydata na nabywcę. Kogóż jednak panowie fachowcy uważają za potencjalną klientelę? Nieodparcie wydaje mi się, że liczą oni głównie na matołów i półgłówków z silnie rozwiniętą potrzebą dowartościowania się.

Nie sposób oprzeć się wrażeniu, że telewizyjne reklamy przedstawiają komputer raczej jako sposób na dowartościowanie się niż jako narzędzie pracy intelektualnej. Szarmancki pan w wytwornym garniturze zasiada na stylowym fotelu przed ekskluzywnym komputerem, gdzie instruktażu udziela mu elegancka pani. Nastrojowa muzyka i sączący się z głośnika kuszący głos apelują raczej do instynktów niż do rozumu i bardziej pasują do reklamy np. perfum niż narzędzia pracy umysłowej, jakim jest komputer. Racjonalnych, merytorycznych przesłanek na rzecz wyboru tego właśnie a nie innego sprzętu raczej nie uświadczysz — zastępuje je, chyba zresztą skutecznie, zwyczajny bajer. Jakie to przekonywające, konkretne argumenty znajdujemy w natrętnej reklamie? „... Doskonałe parametry robocze, bogate oprogramowanie rozwiąże wszystkie Twoje problemy, najlepszy, najdoskonalszy ..."

Nie twierdzę bynajmniej, że uważam telewizyjne reklamy za złe. Wręcz przeciwnie. Martwi mnie jedynie ów opracowany przez speców od reklamy model statystycznego klienta, w którego adekwatność nie mam podstaw wątpić.

Luksusowy samochód, gustowna dyplomatka i ładna kobieta apelują do wyobraźni, gdyż są niewątpliwymi atrybutami pewnego statusu, uważanego potocznie za atrakcyjny. Kłopot tylko w tym, że w odróżnieniu od mercedesa, komputer nie tylko manifestuje materialną zasobność właściciela, ale może też bezlitośnie obnażyć jego intelektualne ubóstwo. Przyjrzyjmy się oto obrazowi wytwornej pani reklamującej komputer, której paluszki z nieporadnym wdziękiem usiłują przebierać po klawiaturze. Głęboko uduchowiony wyraz jej twarzy kazałby widzieć ją raczej za kontuarem, przy kawiarnianym stoliku, no, może przy ręcznych robótkach, ale zapewne nie przy komputerze jako osobistym narzędziu pracy.

Potok mniej lub bardziej bałamutnych informacji płynie zresztą nie tylko z telewizji. Jeden z moich znajomych, zafascynowany wynikami opublikowanego w jednym z popularnych czasopism informatycznych testu mikrokomputera, postanowił nabyć to cudo dla swego zakładu pracy. Po pewnym czasie wytęsknione arcydzieło sztuki inżynierskiej stanęło na jego biurku. Od tej chwili zaczął się szybki proces odbrązowiania cud-maszyny. Niechlujna i niekompletna dokumentacja (a właściwie to, co miało służyć za jej namiastkę), brak programu obsługi dla dostarczonej wraz z komputerem myszki, pozbawiony kontrastu monitor, stochastyczne, lecz dość częste błędy dyskowe przy uruchamianiu systemu, itd., itp. Pierwszym i poniekąd naturalnym odruchem jest pytanie, jak to możliwe, że kopia wyrobu, który otrzymał tak dobrą cenzurkę recenzenta, okazuje się przy bliższym poznaniu zwyczajnym rupieciem. Czyżby po prostu pech? Niekoniecznie. Jednym z możliwych wytłumaczeń może być fakt, że udostępniany testującym egzemplarz bywa zazwyczaj wcześniej szczególnie starannie dopracowany i przetestowany.

Nie chcę przez to powiedzieć, że uważam publikowane w prasie testy i omówienia za nieprawdziwe i niepotrzebne. Przeciwnie, zawierają one zwykle mnóstwo interesujących i cennych informacji. Przestrzegałbym tylko przed traktowaniem samego faktu opublikowania wyników testu sprzętu i wydania mu pozytywnej opinii jako swoistego certyfikatu gwarancyjnego. Ostrożność jest wskazana w każdym przypadku. Uważnie czytając sprawozdanie można czasem między wierszami znaleźć niejedną cenną wskazówkę. Aby ją wykorzystać we własnej praktyce, trzeba jednak zawezwać na pomoc własne oczy i uszy, wspomagane wyobraźnią, krytycyzmem i zdrowym rozsądkiem.

Nie sposób czasem nie odnieść wrażenia, że niektóre firmy komputerowe, zabiegające o klienta w szeroko prowadzonej kampanii reklamowej, stawiają raczej na ilość, niż na jakość. Zwyczajny bajer jest tym skuteczniejszy, że świadomość potencjalnych użytkowników wciąż jeszcze jest niska, a oprócz tego decyzję o zakupie nie zawsze podejmują ci najbardziej kompetentni. Oto zabawny przykład. Kilku pracowników pewnej instytucji zostało oddelegowanych na kilkutygodniowy kurs, obejmujący zasady obsługi komputerów osobistych, standardowe oprogramowanie, tajniki wyboru optymalnej konfiguracji i osprzętu do różnych zastosowań. Aczkolwiek dla kursantów był to pierwszy poważniejszy kontakt z mikrokomputerem, dzięki rzetelnemu podejściu do zajęć zdobyli na kursie solidne podstawy. Jakież było zdziwienie tej grupki, gdy po powrocie do macierzystej firmy zastali świeżo zakupiony za ciężkie pieniądze komputer-bubel, zestawiony z przestarzałych pakietów i w bezsensownej dla ich potrzeb konfiguracji. Dobroduszny szef postanowił bowiem sprawić swym podwładnym niespodziankę i zakupił sprzęt, który „wcisnęła" mu w międzyczasie jakaś przedsiębiorcza firma.

Poważnym problemem naszego rynku komputerowego jest wciąż brak małych, niezależnych firm świadczących kompleksowe usługi z zakresu obiektywnego doradztwa technicznego, jak również brak uświadomionej potrzeby skorzystania z takich usług ze strony potencjalnych nabywców komputerów. Doradztwo świadczone przez dostawców sprzętu w praktyce sprowadza się głównie do zachwalania własnych wyrobów. Konkurencja na rynku komputerowym jest spora, ale [illegible] wiele z tego wynika, gdyż wielu klientów nie próbuje nawet rzetelnie porównywać konkurencyjnych ofert od strony merytorycznej. Jeśli już, to porównuje się tylko cenniki

Dopóki potencjalny klient będzie pozostawiony pod reklamowym ostrzałem bez armat, bez mapy i bez nadziei na fachową odsiecz, dopóty pozostanie mu albo rejterada, albo desperacka szarża na oślep, w stronę, z której dobiega najgłośniejszy harmider. Jeśli idzie o szarże, to mamy tu niewątpliwie chwalebne tradycje. Nasze tradycje w dziedzinie przemyślanych, trafnych inwestycji i efektywnego gospodarowania prezentują się, niestety, znacznie mniej imponująco...

Roland Wacławek



UWAGA CZYTELNICY!

Redakcja kwartalnika „MŁODY TECHNIK — InforMik" bardzo przeprasza Was za opóźnienie ukazania się w kioskach nr. 1/88 naszego pisma.

Jednocześnie zapraszamy Was do dyskusji nad zakresem „InforMika" na zasadzie oceny tego, co już wydrukowaliśmy i sugestii tematycznych na przyszłość.

Czekamy na Wasze listy!

Redakcja „InforMika"



młody TECHNIK
InforMik

MŁODY TECHNIK — INFORMIK
MAGAZYN KOMPUTEROWY
„MŁODEGO TECHNIKA"
NR II (6) ROCZNIK II

SPIS TREŚCI

Zdjęcia w numerze: Władysław P. Jabłoński, Grzegorz Zalot, Jacek Nowicki.
Rysunki: Jerzy Flisak, Roman Gaik.



W szponach reklamy

Pewien mój znajomy zdziwił się niepomiernie przeglądając pierwsze numery „InforMika" faktem, że nie publikujemy reklam. Przecież — jego zdaniem — jest to świetny sposób zarobienia na doskonały sprzęt komputerowy dla redakcji, a zresztą „wszyscy tak robią". Obawiam się jednak czy my, to znaczy redakcja „InforMika", chcemy być tacy jak owi „wszyscy". Po pierwsze: uważamy, że nasi Czytelnicy płacą za informacje zawarte w numerze, a nie za zamieszczone w nim reklamy. Poważne czasopisma zachodnie, jeśli zamieszczają — przykładowo — trzy strony płatnych reklam to o tyle samo stron rozszerzają objętość numeru. My, ze względu na naszą sytuację papierowo-poligraficzną, czynić podobnie nie możemy, a inaczej nie uważamy za stosowne.

Drugą przyczyną naszej niechęci do reklamy jest to, że nie sposób reklamować jakiegoś wyrobu z branży komputerowej bez znajomości potrzeb osoby zainteresowanej kupnem, bowiem to komputer trzeba dostosować do konkretnego zadania. Nie ma np. sensu kupować najnowocześniejszego komputera, jeśli będzie on służył tylko jako maszyna do pisania. Aby coś reklamować w sposób fachowy trzeba znać bardzo wiele aspektów kupna, np. zastosowanie komputera, typ oprogramowania, możliwości finansowe kontrahenta itp. itd. Istnieje cały szereg poważnych firm, które zajmują się doradztwem przy zakupie sprzętu informatycznego, firm wyspecjalizowanych i bardzo dbających, aby klient za swoje pieniądze otrzymał maksymalnie dobry sprzęt i oprogramowanie dla swoich potrzeb przy możliwie jak najniższym wydatku. A więc i do informatyki wkroczył rachunek ekonomiczny, i to już od dawna, bo kto jak kto, ale „komputerowcy" liczyć umieją...

Gdyby natomiast ktoś nam zarzucił, że przecież w sumie i tak reklamujemy te rzeczy spośród hardware'owej maszynerii, o której piszemy, to mamy na swoje usprawiedliwienie tylko to, że piszemy głównie o rzeczach dobrych, tanich i sprawdzonych dla informatyków-hobbistów lub o nowych konstrukcjach, o których warto wiedzieć choćby dlatego, aby mieć pojęcie o tym, co dzieje się w branży informatycznej na świecie. Nasze testy są również sporządzane pod kątem przydatności danego urządzenia do praktyki hobbistycznej.

Nasza rodzima reklama, zaniedbana przez dziesięciolecia, czyni na razie pierwsze nieporadne kroki, złoszcząc czasem swą naiwną i powierzchowną formą. Jest na razie bardziej sygnałem o istnieniu tego czy owego niż informacją przeznaczoną dla fachowego odbiorcy. Póki nie nauczymy się, że technika wymaga argumentów technicznych i ekonomicznych w treści reklamy, musi wystarczyć nam wspólny portret komputera, kozy i ładnej panienki...

JERZY KLAWIŃSKI



ROLAND WACŁAWEK

POLSKIE ZNAKI DIAKRYTYCZNE NA DRUKARCE

Tam, gdzie komputer osobisty nie jest gadżetem, lecz narzędziem pracy, prędzej czy później wynika problem polskich znaków diakrytycznych.

W przypadku ekranu sprawa jest przynajmniej dość jednoznaczna. Albo program wykorzystuje ekran w trybie znakowym (np. WordStar lub dBASE III) i wtedy jedynym wyjściem jest przeróbka karty graficznej (pomijamy tu możliwość definiowania własnych znaków w karcie EGA), albo ekran pracuje w trybie graficznym (np. FRAMEWORK II lub MS-WORD) a więc litery są „przerysowywane" z pewnego zbioru wzorców. W tym drugim przypadku można pokusić się u rozpoznanie i przedefiniowanie wewnętrznego zestawu wzorców w programie, uzyskując polskie litery bez potrzeby adaptacji sprzętu.

Rozwiązanie problemu wydruku polskich liter na drukarce mozaikowej jest w ogólności bardziej złożone. Wynika to zarówno z różnych sposobów, w jakie programy traktują drukarki, jak również z mnogości różnych typów drukarek i ich różnych właściwości. Oto kilka typowych rozwiązań: sprzętowa przeróbka drukarki, wykorzystanie możliwości programowego definiowania znaków (ang. downloading), „dorysowywanie" polskich znaków w trybie graficznym, przetwarzanie na postać graficzną całych wierszy i „składanie" znaków, tzn. uzyskiwanie polskich liter przez nadrukowanie kilku znaków w tym samym miejscu.

Najlepsze rezultaty daje na ogół przeróbka sprzętowa, polegająca na wymianie pamięci stałej, zawierającej wzorce znaków, na podobną pamięć np. EPROM, w której część mniej potrzebnych symboli standardowego, rozszerzonego kodu IBM-ASCII wymieniono na polskie litery. Tak przerobiona drukarka zachowuje pełną szybkość pracy niezależnie od tego, czy polskie litery występują w tekście, czy też nie, jest też odporna na wszelkie niesprzyjające okoliczności, o których później. Mankamentem jest natomiast konieczność fizycznej ingerencji do wnętrza drukarki oraz konieczność związania się na stałe z jedną z wielu stosowanych w Polsce konwencji kodowania polskich liter.

Standardowy kod ASCII zawiera 128 znaków o kodach od 0 do 127. Oprócz cyfr i liter alfabetu angielskiego obejmuje on znaki interpunkcyjne i liczne specjalne znaki sterujące dla drukarek. W sprzęcie klasy IBM-PC/XT/AT znalazł zastosowanie tzw. rozszerzony kod ASCII, w którym występuje 256 znaków o kodach od 0 do 255. Pierwszych 128 znaków jest identycznych ze znakami standardowego kodu ASCII, pozostałe znaki o kodach od 128 do 255 przedstawiają znaki diakrytyczne niektórych języków europejskich (niemieckie, francuskie, szwedzkie itd.) oraz specjalne symbole graficzne: litery greckie, symbole matematyczne, elementy

ramek itd. W zestawie tym nie ma jednak liter specyficznie polskich, jak ą, ę itd.

Aby uzyskać możliwość operowania polskimi literami trzeba wprowadzić je w miejsce wybranych znaków rozszerzonego kodu ASCII. Ponieważ elementy ramek i inne symbole graficzne są wykorzystywane w wielu zastosowaniach, w praktyce pozostaje zastąpienie wybranych znaków języków europejskich. Ponieważ język polski liczy 18 znaków diakrytycznych (9 liter małych i tyleż dużych), trzeba zrezygnować z takiej samej liczby pierwotnych znaków rozszerzonego kodu ASCII. Ponieważ dotychczas nie ma obowiązującej normy kodowania polskich liter w rozszerzonym kodzie ASCII (stan z 30 grudnia 1987), w kraju spotykanych jest wiele różnych systemów kodowania. Dużą popularnością cieszą się dwa: kod wprowadzony przez firmę CSK (Computer Studio Kajkowscy) oraz kod stosowany w komputerach MAZOVIA, przejmowany ostatnio przez coraz większą liczbę niezależnych dostawców sprzętu.

Rozmaite są przesłanki, przyświecające projektantom różnych kodów. Jednym razem jest to chęć zachowania stałej różnicy kodów liter małych i dużych, identycznej jak w przypadku liter alfabetu angielskiego (np. tzw. kod CSK). Kiedy indziej chodzi o zachowanie maksymalnej czytelności tekstu nawet na nie przerobionej karcie graficznej (np. tzw. kod MAZOVII). Nie brak oczywiście i innych kodów, w których trudno doszukać się jakiejkolwiek rozsądnej tendencji poza dążeniem do maksymalnej prostoty (porównaj kod stosowany m.in. przez firmę DHN). Trzy przykładowe systemy kodowania podano w tabeli 1. Po lewej podano kod w postaci dziesiętnej, po prawej, w nawiasach okrągłych — w formie szesnastkowej.

W związku z mnogością kodów i kłopotami towarzyszącymi przeróbkom sprzętowym w tym artykule skoncentrujemy się na uzyskiwaniu polskich liter sposobami programowymi, bez przeróbek drukarki.

Ponieważ większość dostępnych obecnie na rynku drukarek dysponuje możliwością programowego definiowania własnych znaków, to wykorzystanie tej możliwości wydaje się najprostszym sposobem uzyskania w wydruku polskich liter. Niestety, niewiele jest drukarek, w których na znaki definiowane programowo nie nakłada się jakichś dotkliwych restrykcji — dotyczą one zwłaszcza sprzętu popularnego. A to liczba definiowanych znaków jest ograniczona np. do 96, a to wysokość znaku nie może przekroczyć 8 punktów rastra (podczas gdy znaki standardowe mogą mieć wysokość do 9 punktów), a to przełączanie w jednej linii między znakami standardowymi i definiowanymi nie jest możliwe i w najlepszym razie prowadzi do drastycznego ograniczenia wydajności, a to nie można zdefiniować znaków w trybie NLQ, a to dla znaków zdefiniowane programowo nie mogą otrzymać

```
;****************************** POLODIAK ******************************
;>>>>>>>>>  Roland Wacławek,  Siemianowice Śl,  9.12.1986  <<<<<<<<

Kod_prog     SEGMENT

MS_DOS       EQU     21H   ;numer przerwania wywołujacego funkcje DOS
Drukarka     EQU     17H   ;numer przerwania  obsługi drukarki (BIOS)
CzytajWekt   EQU     35H   ;funkcja MS-DOS: odczyt wektora przerwania
Ustaw_Wekt   EQU     25H   ;funkcja MS-DOS: zapis  wektora przerwania
Zintegruj    EQU     27H   ;nr przerwania  konczącego i integrujacego
Kod_ESC      EQU     1BH   ;kod znaku ESCAPE

             ASSUME  CS:Kod_prog, DS:Kod_Prog

             ORG     256

Inicjacja:   JMP     Zainstaluj

Print_Adr:   DW      0     ;offset   pierwotnego   handlera  drukarki
Print_Segm:  DW      0     ;adres segm. pierwotnego handlera drukarki

Status:      DB      0     ;bajt statusowy programu POLODIAK:
                           ;bit 7 = 1: trwa obsługa sekwencji  ESCAPE
                           ;bit 6 = 1: trwa obsługa argumentu  ESCAPE
                           ;bit 5 = 1: ostatni bajt sekwencji  ESCAPE
                           ;bit 4 = 1: przedostatni bajt sekw. ESCAPE
                           ;bit 2 = 1: obsługa  definicji tabulatorow
                           ;bit 0 = 1: tryb graficzny

Licznik:     DW      0     ;licznik bajtow sekwencji  trybu graficzn.

BajtPost:    AND     BYTE PTR Status, 11101111B
             MOV     BYTE PTR Licznik, AL
             JMP     SkokOutput

BajtOstat:   MOV     BYTE PTR Licznik+1, AL        ;wstepnie zapamiętaj
             TEST    BYTE PTR Status, 1 ;czy obsługa sekw. graficzn.?
             JZ      Kesc               ;nie -Koniec sekwencji ESCAPE
             AND     WORD PTR Licznik, 0FFFFH ;to jest tryb graficzny
             TEST    WORD PTR Licznik, 0FFFFH ;testuj, czy licznik =0
             JNZ     Kesc                     ;tak - Koniec sekwencji
             AND     BYTE PTR Status, 11111100B  ;zeruj bit trybu gr.
Kesc:        JMP     Koniec_ESC     ;zakoncz obsługe sekwencji ESCAPE

Obslug_ESC:  TEST    BYTE PTR Status, 10H  ;czy ost.  bajt sekw. ESC?
             JNZ     BajtPost              ;nie -  3 z czworki bajtow
             TEST    BYTE PTR Status, 20H  ;przedost. bajt sekwencji?
             JNZ     BajtOstat             ;tak - zapamiętaj ten bajt
             TEST    BYTE PTR Status, 4H   ;czy definicja tabulatora?
             JZ      Koniec_ESC            ;nie -  Koniec obsługi ESC
             AND     AL, AL                ;test, czy 0 =sygnał Konca
             JNZ     SkokOutput            ;nie -prześlij do drukarki
             JMP     Koniec_ESC            ;tak-Koniec definicji tab.

Przedostat:  AND     BYTE PTR Status, 11011111B  ;Kasuj flage bajtu 3
             MOV     BYTE PTR Licznik, AL   ;zapamiętaj  młodszy bajt
             JMP     SkokOutput             ;i  wyślij go do drukarki

Inicjuj:     MOV     CS:BYTE PTR Status, 0 ;wyzeruj  bajt  statusowy
             MOV     CS:WORD PTR Licznik, 0 ;wyzeruj  licznik  bajtow
             JMP     Do_BIOS                ;inicjacja sprzętowa  dr.

Transform:   CMP     AH, 1     ;tu po przerwaniu 23-test, czy inicjacja
             JZ      Inicjuj ;tak - zainicjuj ten program i drukarkę
             CMP     AH, 0     ;czy  funkcja normalnego wydruku znakow?
             JZ      Drukuj_Zn  ;tak, funkcja numer 0: wydrukuj znak
Do_BIOS:     JMP     CS:DWORD PTR Print_Adr ;skok do pierw. handlera

Drukuj_Zn:   TEST    CS:BYTE PTR Status, 80H ;czy tryb obsługi ESCAPE
             JZ      Tryb_Norm            ;nie - normalna analiza znaku
             PUSH    DS                   ;przechowaj uzywane  rejestry
             PUSH    SI                   ;ten  fragment   interpretuje
             PUSH    BX                   ;sekwencje znakow  rozpoczęte
             PUSH    CX                   ;kodem sterujacym ESCAPE =1BH
             PUSH    CS                   ;niech segment  danych stanie
             POP     DS                   ;się zgodny z  segmentem kodu
             TEST    BYTE PTR Status, 40H  ;czy pierwszy znak po ESC?
             JZ      Obslug_ESC            ;jezeli nie,   obsługa ESC
Pierwszy:    AND     BYTE PTR Status, 10111111B  ;kasuj bit argumentu
             MOV     BX, OFFSET Tabl_ESC      ;adres tablicy kodow ESC
             MOV     CL, (OFFSET AdrTabl-OFFSET Tabl_ESC)/3 ;il. elem.
Nastepny:    CMP     AL, [BX]             ;kod znaku zgodny ze wzorcem?
             JE      Znalez_Kod           ;tak - zakoncz przeszukiwanie
             ADD     BX, 3                ;ustaw adres kolejnego wzorca
             DEC     CL                   ;CL=licznik elementow tablicy
             JNZ     Nastepny             ;gdy nie koniec, szukaj dalej
```

```
Koniec_ESC: AND     BYTE PTR Status,00000001B ;kasuj bity trybu ESC
SkokOutput: JMP     Output_LPT          ;wyslij znak z AL na drukarke

Znalez_Kod: JMP     [BX+1]              ;skocz  do  adresu  z tablicy

Grafika_4B: OR      BYTE PTR Status, 00000001B ;ustaw flage grafiki
Cztery_Bty: OR      BYTE PTR Status, 10110000B ;ustaw bity trybu 4B
            JMP     SkokOutput          ;wyprowadz  bajt  na drukarke

Trzy_Bajty: OR      BYTE PTR Status, 10100000B ;ustaw bity trybu 3B
            JMP     SkokOutput

Tabulator:  OR      BYTE PTR Status, 10000100B ;ustaw bity tabulat.
            JMP     SkokOutput          ;wyprowadz  bajt  na drukarke

            ASSUME  CS:Kod_prog, DS:Nothing

Tryb_Norm:  TEST    BYTE PTR Status,00000001B   ;czy to tryb graf.?
            JNZ     Tryb_Graf           ;tak -  to bajt bloku grafiki
            CMP     AL,Kod_ESC          ;czy  to  jest  znak  ESCAPE?
            JE      Escape              ;jezeli tak,  wlacz tryb  ESC
            TEST    AL,10000000B        ;kod znaku  wiekszy  od  127?
            JNZ     Konwersja           ;tak-moze konieczna konwersja
Wyprowadz:  JMP     DWORD PTR Print_Adr ;skocz do pierwot. handlera

Tryb_Graf:  DEC     WORD PTR Licznik  ;odlicz   wyprowadzany   bajt
            JNZ     Wyprowadz           ;jesli nie ostatni, wyprowadz
            AND     BYTE PTR Status,11111110B   ;kasuj bit grafiki
            JMP     Wyprowadz           ;wyprowadz  bajt  do drukarki

Escape:     MOV     BYTE PTR Status, 11000000B ;ustaw bit trybu ESC
            JMP     Wyprowadz           ;wyslij znak ESC  do drukarki

            ASSUME  CS:Kod_prog, DS:Kod_Prog

Konwersja:  PUSH    DS                  ;przechowaj  uzywane rejestry
            PUSH    SI
            PUSH    BX
            PUSH    CX
            PUSH    CS                  ;skopiuj zawartosc  CS do  DS
            POP     DS
            XOR     BX,BX               ;zeruj zawartosc rejestru  BX
            MOV     SI, OFFSET AdrTabl  ;SI=adres tablicy konwersji
Badaj_Nast: CMP     BYTE PTR[SI][BX],00 ;czy to juz koniec tablicy?
            JZ      Output_LPT          ;tak -  kodu nie znaleziono
            CMP     [SI][BX],AL         ;porownaj kod znaku z tabl.
            JZ      Znaleziony          ;jesli zgodny, to konwersja
            INC     BX                  ;ustaw w BX  wsk. nast. znaku
            JMP     Badaj_Nast          ;porownaj  z  nast. elementem
Znaleziony: ADD     BX,BX               ;pomnoz zawartosc BX  przez 8
            ADD     BX,BX
            ADD     BX,BX
            MOV     SI, OFFSET Tabl_podm ;SI =adres tablicy zamiany
Ptla_pdmn:  MOV     AL,[SI][BX]         ;AL = kolejny bajt sekw. zam.
            OR      AL, AL              ;czy jest to  wskaznik konca?
            JZ      Koniec_wyp          ;tak -  koniec sekwencji zam.
            XOR     AH,AH               ;AH=kod funkcji wydruku znaku
            PUSHF                       ;skoryguj stos dla rozk. IRET
            CALL    DWORD PTR Print_Adr ;wywolaj pierwotny  handler
            INC     BX                  ;przesun wskaznik na nast. zn
            JMP     Ptla_pdmn           ;powtorz dla nastepnego znaku

Output_LPT: MOV     AH,0                ;AH=kod funkcji: emisja znaku
Wyprow_LPT: PUSHF                       ;skoryguj stos dla rozk. IRET
            CALL    DWORD PTR Print_Adr ;wywolaj pierwotny  handler
Koniec_Wyp: POP     CX                  ;odtworz  uzywane  rejestry
            POP     BX
            POP     SI
            POP     DS
            IRET                        ;powrot  z obslugi przerwania

Tabl_ESC    DB      'X'
            DW      OFFSET Cztery_Bty
            DB      '?'
            DW      OFFSET Cztery_Bty
            DB      '*'
            DW      OFFSET Trzy_Bajty
            DB      '/'
            DW      OFFSET Trzy_Bajty
            DB      '3'
            DW      OFFSET Trzy_Bajty
            DB      '-'
            DW      OFFSET Trzy_Bajty
```

niektórych atrybutów, itd., itp. Za typowy przykład powyższych ograniczeń mogą posłużyć szeroko rozpowszechnione drukarki STAR GEMINI Xi, SG lub NL.

Korzystanie ze znaków definiowanych programowo prowadzi do kłopotów także wtedy, gdy przez rozłączenie lub np. z powodu zablokowania papieru wyłączymy drukarkę podczas pracy, co powoduje na ogół utratę załadowanych do drukarki definicji znaków (większość drukarek nie dysponuje bateryjnym podtrzymaniem zawartości pamięci definiowanych znaków). Oprócz tego niektóre programy wysyłają do drukarki po załadowaniu lub przed rozpoczęciem wydruku sekwencję kodów inicjujących, które na ogół uaktywniają ponownie standardowy, tzn. stały generator znaków.

„Dorysowywanie" polskich liter w trybie graficznym może być realizowane przez program rezydujący, na stałe zainstalowany w pamięci operacyjnej, który przechwytuje wszelkie informacje przeznaczone dla drukarki, rozpoznaje kody zarezerwowane dla polskich znaków i zamiast nich wysyła do drukarki odpowiednią kompozycję graficzną. Rozwiązanie to jest o tyle wygodne, że ani program, ani drukarka „nie widzą" dodatkowego pośrednika, zainstalowanego między nimi, zaś ewentualne włączanie i wyłączanie drukarki także nie powoduje kłopotów. Problemem może być natomiast uzyskanie pełnej wysokości znaków (nie wszystkie drukarki pozwalają na graficzny druk 9 igłami). Poza tym, ponieważ znaki drukowane w trybie graficznym występują na przemian ze znakami standardowymi, trzeba zadbać o dokładne dopasowanie krojów znaków. Niestety, prawie każda drukarka różni się niuansami krojów pisma od innych. Oprócz tego, jeżeli drukarka dysponuje z natury kilkoma różnymi krojami pisma, pismem korespondencyjnym itd., to dla każdego z nich trzeba zdefiniować odrębne wzorce znaków. Trzeba też zaprogramować sposób uzyskiwania takich mutacji znaków, jak kursywa czy pogrubienie. W trybie graficznym nie można uzyskać niektórych atrybutów, jak np. podwójne uderzenie. Łączenie trybów: tekstowego i graficznego w jednej linii prowadzi też niekiedy do nieoczekiwanych problemów, np. wtedy, gdy w linii tej występują specjalne znaki semigraficzne o wysokości półtorej linii (kody 176 .. 223).

Najbardziej uniwersalnym i radykalnym rozwiązaniem jest zastosowanie programu rezydującego, przechwytującego wszystkie znaki wysyłane do drukarki i po zakończeniu każdej linii przetwarzającego je w komplecie na postać graficzną. W tym przypadku wewnętrzny generator znaków nie jest wykorzystywany wcale. Zaletą tego wariantu jest zupełne uniezależnienie się od typu drukarki i możliwość uzyskania wydruku o znacznie lepszej aparycji niż pozwala na to sama drukarka, np. jakości korespondencyjnej i kursywy na drukarkach bez tych atrybutów pisma. Ceną jest zwolnienie tempa wydruku i pewne

```
          DB        'A'
          DW        OFFSET Trzy_Bajty
          DB        'B'
          DW        OFFSET Tabulator
          DB        'a'
          DW        OFFSET Trzy_Bajty
          DB        'C'
          DW        OFFSET Trzy_Bajty
          DB        'D'
          DW        OFFSET Tabulator
          DB        'h'
          DW        OFFSET Trzy_Bajty
          DB        'i'
          DW        OFFSET Trzy_Bajty
          DB        'J'
          DW        OFFSET Trzy_Bajty
          DB        'j'
          DW        OFFSET Trzy_Bajty
          DB        'K'
          DW        OFFSET Grafika_4B
          DP        'L'
          DW        OFFSET Grafika_4B
          DB        'l'
          DW        OFFSET Trzy_Bajty
          DB        'N'
          DW        OFFSET Trzy_Bajty
          DB        'Q'
          DW        OFFSET Trzy_Bajty
          DB        'p'
          DW        OFFSET Trzy_Bajty
          DB        'R'
          DW        OFFSET Trzy_Bajty
          DB        'r'
          DW        OFFSET Trzy_Bajty
          DB        'S'
          DW        OFFSET Trzy_Bajty
          DB        'U'
          DW        OFFSET Trzy_Bajty
          DB        'W'
          DW        OFFSET Trzy_Bajty
          DB        'x'
          DW        OFFSET Trzy_Bajty
          DB        'Y'
          DW        OFFSET Grafika_4B
          DB        'Z'
          DW        OFFSET Grafika_4B

Adrtabl:  DB    134, 141, 145, 146, 164, 162, 158, 166, 167
          DB    143, 149, 144, 156, 165, 163, 152, 160, 161
          DB    0

Tabl_Podm: DB  'a',   8, ',', 000, 000, 000, 000, 000      ;a,
          DB   'c',   8, "'", 000, 000, 000, 000, 000      ;c'
          DB   'e',   8, ',', 000, 000, 000, 000, 000      ;e,
          DB   'l',   8, '/', 000, 000, 000, 000, 000      ;l/
          DB   'n',   8, "'", 000, 000, 000, 000, 000      ;n'
          DB   'o',   8, "'", 000, 000, 000, 000, 000      ;o'
          DB   's',   8, "'", 000, 000, 000, 000, 000      ;s'
          DB   'z',   8, '^', 000, 000, 000, 000, 000      ;z'
          DB   'z',   8, "'", 000, 000, 000, 000, 000      ;z.

          DB   'A',   8, ',', 000, 000, 000, 000, 000      ;A,
          DB   'C',   8, "'", 000, 000, 000, 000, 000      ;C'
          DB   'E',   8, ',', 000, 000, 000, 000, 000      ;E,
          DB   'L',   8, '/', 000, 000, 000, 000, 000      ;L/
          DB   'N',   8, "'", 000, 000, 000, 000, 000      ;N'
          DB   'O',   8, "'", 000, 000, 000, 000, 000      ;O'
          DB   'S',   8, "'", 000, 000, 000, 000, 000      ;S'
          DB   'Z',   8, '^', 000, 000, 000, 000, 000      ;Z'
          DB   'Z',   8, '-', 000, 000, 000, 000, 000      ;Z.
Zainstaluj: MOV    AH, CzytajWekt              ;wywolaj funkcje odczytu
          MOV      AL, Drukarka                ;wektora przerwania 17H
          INT      MS_DOS                      ;odczytaj wektor do ES:BX
          MOV      WORD PTR Print_Adr, BX      ;przechowaj segment+offs.
          MOV      WORD PTR Print_Segm, ES     ;wektora przerwania 17H
Wektory:  MOV      DX, OFFSET Transform
          MOV      AH, Ustaw_Wekt
          MOV      AL, Drukarka
          INT      MS_DOS                      ;zapisz nowy wektor 17H
          MOV      DX, OFFSET Zainstaluj
          INT      Zintegruj                   ;uczyn program rezydujacym

Kod_Prog  ENDS
          END      Inicjacja
```

uszczuplenie pamięci operacyjnej. Musi ona bowiem pomieścić definicje kompletu wzorców jednego lub kilku krojów pisma oraz specjalny, dość skomplikowany program drukujący. Właściwości samej drukarki są w tym przypadku niezbyt istotne — w końcu pracuje ona tylko jako maszyna do „tłuczenia" punktów. Ważna jest tylko odpowiednia rozdzielczość graficzna. W ten sposób działa m.in. program LETTRIX, zaś niektóre edytory wyprowadzają swoje wyniki od razu w trybie graficznym (np. CHIWRITER lub niektóre programy działające w systemie WINDOWS).

Stosunkowo najprostszym w realizacji, uniwersalnym i zapewniającym niezłe efekty sposobem uzyskania polskich znaków jest drukowanie ich przez kolejne nakładanie (nadrukowywanie) kilku znaków. Tak np. literę 'ą' można uzyskać jako złożenie litery 'a' i przecinka, natomiast 'Ł' poprzez złożenie litery 'L' i ukośnika '/'. W najprostszym przypadku nadrukowywanie wymaga możliwości cofania głowicy drukarki o jedno pole znakowe, co realizuje znak sterujący (BACKSPACE) (ASCII 8). Znak ten jest akceptowany przez większość drukarek, aczkolwiek nie przez wszystkie.

W niniejszym artykule postaramy się zaprezentować sposób realizacji prostego programu rezydującego, transformującego dane wysyłane do drukarki. Nasz przykładowy program POLODIAK realizuje wydruk polskich znaków metodą nadrukowywania znaków z cofaniem głowicy. Może on obsługiwać praktycznie każdą drukarkę, zgodną ze standardem EPSON FX-80 lub IBM-GP z możliwością realizacji funkcji (BACKSPACE) (wymaganie to spełniają m.in. popularne STAR SG-10/15, STAR NL-10 i STAR NX-10/15). Opisany program może współpracować z prawie każdym typowym oprogramowaniem, jak dBASE III+, WordStar itd.

Program POLODIAK wykorzystuje fakt, że praktycznie wszystkie programy wysyłają dane do drukarki za pośrednictwem przerwania programowego nr 17H (dziesiętnie: 23), którego procedura obsługi znajduje się w pamięci BIOS-ROM. Przerwanie to realizuje 3 funkcje o kodach 0-2. W chwili spowodowania przerwania rozkazem INT 17H kod funkcji powinien znajdować się w rejestrze AH, natomiast ewentualny kod znaku do wyprowadzenia — w rejestrze AL. AH=0 oznacza wyprowadzenie znaku na drukarkę, AH=1 — inicjację drukarki, wreszcie AH=2 — odczyt bieżącego statusu drukarki (ta ostatnia funkcja nas nie interesuje). Oprócz tego rejestr DX zawiera numer interfejsu drukarki, której operacja dotyczy.

Po uruchomieniu program POLODIAK wykonuje tylko trzy czynności: zapamiętuje starą wartość wektora przerwania programowego nr 17H (23), wskazującego normalnie odpowiednią procedurę BIOS-ROM, wpisuje w jego miejsce nowy wektor, wskazujący na etykietę **Transform**, po czym instaluje na stałe w pamięci nową procedurę obsługi przerwania 17H.

```
{Program ladujacy  POLODIAK}
{Roland Waclawek 12.02.1988}
CONST dane:ARRAY[0..$247] OF Byte=
  ($E9,$29,$02,$00,$00,$00,$00,$00,
   $00,$00,$80,$26,$07,$01,$E7,$A2,
   $06,$01,$E9,$95,$00,$A2,$09,$01,
   $F6,$06,$07,$01,$01,$74,$13,$81,
   $26,$06,$01,$FF,$0F,$F7,$06,$08,
   $01,$FF,$FF,$75,$05,$80,$26,$07,
   $01,$FC,$EB,$71,$90,$F6,$06,$07,
   $01,$10,$75,$CE,$F6,$06,$07,$01,
   $20,$75,$D2,$F6,$06,$07,$01,$04,
   $74,$5B,$22,$C0,$75,$5C,$EB,$56,
   $90,$80,$26,$07,$01,$DF,$A2,$08,
   $01,$EB,$4F,$90,$2E,$C6,$06,$07,
   $01,$00,[illegible],$C7,$06,$08,$01,$00,
   $00,$EB,$0B,$90,$80,$FC,$01,$74,
   [illegible],$80,$FC,$00,$74,[illegible],$2E,$FF,
   $2E,$03,$01,$2E,$F6,$06,$07,$01,
   $80,$79,$47,$1E,$56,$53,$51,$0E,
   $1F,$F6,$06,$07,$01,$40,$74,$A5,
   $80,$26,$07,$01,$BF,$EB,$32,$02,
   $B1,$1D,$3A,$07,$74,$0F,$83,$C3,
   $03,$FE,$C9,$75,$F5,$80,$26,$07,
   $01,$01,$EB,$7A,$90,$FF,$67,$01,
   $80,$0E,$07,$01,$01,$80,$0E,$07,
   $01,$80,$EB,$EE,$80,$0E,$07,$01,
   $A0,$EB,$E7,$80,$0E,$07,$01,$04,
   $EB,$E0,$2E,$F6,$06,$07,$01,$01,
   $75,$0D,$3C,$1B,$74,$16,$A6,$80,
   $75,$1C,$2E,$FF,$2E,$03,$01,$2E,
   $FF,$0E,[illegible],$01,$75,$F4,$2E,$80,
   $26,$07,$01,$FE,$EB,$BC,$2E,$C6,
   $06,$07,$01,$C0,$EB,$E4,$5B,$56,
   $53,$51,$0E,$1F,$33,$DB,$8E,$69,
   $02,$80,$38,$00,$74,$20,$38,$00,
   $74,$03,$43,$EB,$F4,$03,$DB,$03,
   $DB,$03,$DB,$BE,$9C,$02,$8A,$00,
   $0A,$C0,$74,$11,$32,$E4,$9C,$FF,
   $1E,$03,$01,$43,$EB,$F0,$B4,$00,
   $9C,$FF,$1E,$03,$01,$59,$5B,$5E,
   $1F,$CF,$25,$B5,$01,$3F,$B5,$01,
   $7E,$BC,$01,$2F,$BC,$01,$33,$BC,
   $01,$2D,$BC,$01,$41,$BC,$01,$42,
   $C3,$01,$61,$BC,$01,$43,$BC,$01,
   $44,$C3,$01,$68,$BC,$01,$69,$BC,
   $01,$4A,$BC,$01,$6A,$BC,$01,$4B,
   $B0,$01,$4C,$B0,[illegible],$6C,$BC,$01,
   $4E,$BC,$01,$61,$BC,$01,$70,$BC,
   $01,$52,$BC,$01,$72,$BC,$01,$63,
   $BC,$01,$56,$BC,$01,$57,$BC,$01,
   $78,$BC,$01,$59,$B0,$01,$5A,$B0,
   $01,$88,$8D,$91,$92,$A4,$A2,$9E,
   $A6,$A7,$8F,$95,$90,$9C,$A5,$A3,
   $98,$A0,$A1,$00,$61,$08,$2C,$00,
   $00,$00,$00,$00,$63,$08,$27,$00,
   $00,$00,$00,$00,$65,$08,$2C,$00,
   $00,$00,$00,$00,$6C,$08,$2F,$00,
   $00,$00,$00,$00,$6E,$08,$27,$00,
   $00,$00,$00,$00,$6F,$08,$27,$00,
   $00,$00,$00,$00,$73,$08,$27,$00,
   $00,$00,$00,$00,$7A,$08,$7E,$00,
   $00,$00,$00,$00,$7A,$08,$27,$00,
   $00,$00,$00,$00,$41,$08,$2C,$00,
   $00,$00,$00,$00,$43,$08,$27,$00,
   $00,$00,$00,$00,$45,$08,$2C,$00,
   $00,$00,$00,$00,$4C,$08,$2F,$00,
   $00,$00,$00,$00,$4E,$08,$27,$00,
   $00,$00,$00,$00,$4F,$08,$27,$00,
   $00,$00,$00,$00,$53,$08,$27,$00,
   $00,$00,$00,$00,$5A,$08,$7E,$00,
   $00,$00,$00,$00,$5A,$08,$2D,$00,
   $00,$00,$00,$00,$B4,$35,$B0,$17,
   $CD,$21,$89,$1E,$03,$01,$8C,$06,
   $05,$01,$BA,$6C,$01,$B4,$25,$B0,
   $17,$CD,$21,$BA,$2C,$03,$CD,$27);

VAR i, suma : Integer;
    plik    : FILE OF Byte;
BEGIN Assign(Plik,'POLODIAK.COM');
      ReWrite(Plik); suma:= 0;
      FOR i:= 0 TO $247 DO
        BEGIN Write(Plik, dane[i]);
              suma:= suma+dane[i];
        END;
      IF suma <> -22619
         THEN Write('Bledne dane!')
         ELSE Close(Plik)
END.
```

TABELA 1

Litera	Kod CSK	Kod MAZOVII	Kod DHN itd.
Ą	128 (080H)	143 (08FH)	128 (080H)
Ć	129 (080H)	149 (095H)	129 (081H)
Ę	130 (082H)	144 (090H)	130 (082H)
Ł	131 (083H)	156 (09CH)	131 (083H)
Ń	132 (084H)	165 (0A5H)	132 (084H)
Ó	133 (085H)	163 (0A3H)	133 (085H)
Ś	134 (086H)	152 (098H)	134 (086H)
Ź	135 (087H)	160 (0A0H)	135 (087H)
Ż	136 (088H)	161 (0A1H)	136 (088H)
ą	160 (0A0H)	134 (086H)	137 (089H)
ć	161 (0A1H)	141 (08DH)	138 (08AH)
ę	162 (0A2H)	145 (091H)	139 (08BH)
ł	163 (0A3H)	146 (092H)	140 (08CH)
ń	164 (0A4H)	164 (0A4H)	141 (08DH)
ó	165 (0A5H)	162 (0A2H)	142 (08EH)
ś	166 (0A6H)	158 (09EH)	143 (08FH)
ź	167 (0A7H)	166 (0A6H)	144 (090H)
ż	168 (0A8H)	167 (0A7H)	145 (091H)

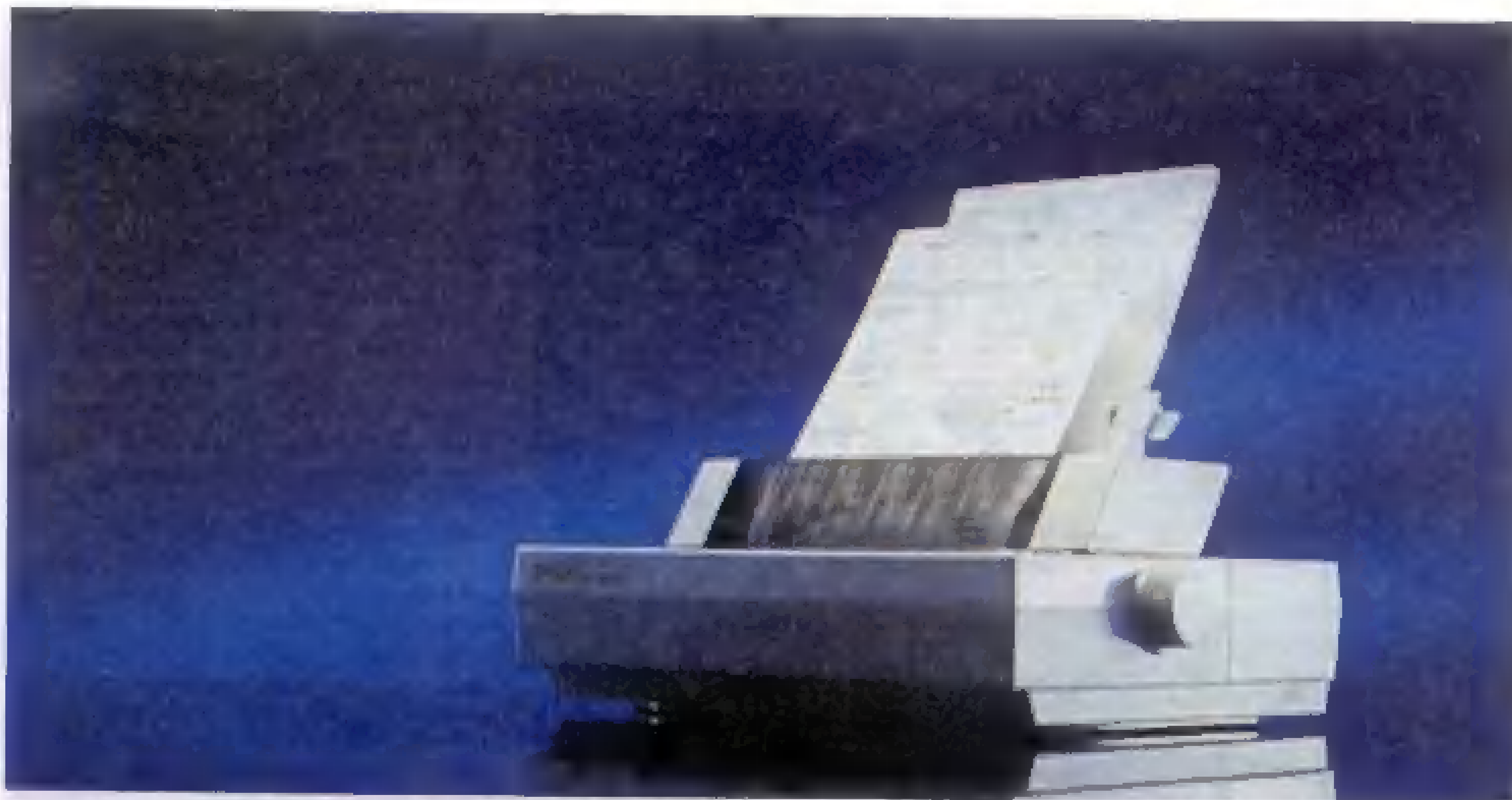
Ta nowa procedura korzysta zresztą z usług starej procedury, poprzez którą wysyłane są do drukarki wszystkie znaki już po ewentualnej transformacji.

Po zainstalowaniu w pamięci nowa procedura obsługi powinna analizować kod każdego znaku wysyłanego do drukarki. Jeżeli nie jest to kod polskiej litery, znak powinien być przesyłany do drukarki bez żadnych modyfikacji. W razie rozpoznania polskiej litery należy zamiast niej wysłać sekwencję złożoną z kilku znaków, rozdzielonych symbolem ⟨BACKSPACE⟩. Np. dla litery 'ą' byłaby to sekwencja: ⟨a⟩ ⟨BACKSPACE⟩ ⟨,⟩.

Zadanie programu byłoby, jak widać, bardzo proste, gdyby nie pewien szkopuł. Otóż oprócz „czystych" tekstów, złożonych wyłącznie ze znaków drukowalnych lub formatujących (np. ⟨CR⟩ i ⟨LF⟩), do drukarki mogą być jeszcze wysyłane informacje sterujące lub dane graficzne. W obydwu przypadkach w strumieniu kierowanych do drukarki bajtów mogą wystąpić kody pokrywające się z kodami polskich liter, lecz tym razem jakakolwiek ich transformacja byłaby nad wyraz niepożądana. Skąd jednak nasz program ma wiedzieć, który bajt zinterpretować jako kod polskiej litery, a który jako np. element wyprowadzonej na drukarkę grafiki?

Jedynym skutecznym rozwiązaniem powyższego problemu jest zorganizowanie w programie bieżącej interpretacji wysyłanych do drukarki danych w taki sam sposób, jak czyni to drukarka. W ten sposób program filtrujący może na bieżąco śledzić stan drukarki i przewidywać jej tryb pracy. Tak np. po rozpoznaniu sekwencji ⟨ESC⟩ ⟨L⟩, zwiastującej dane graficzne,





program powinien przechwycić dwa następne bajty, zawierające długość ciągu graficznego, po czym przepuścić bez dalszej analizy dokładnie tyle bajtów, ile wynosi podana długość. Jak się okazuje, analiza możliwych przypadków szczególnych kosztuje więc nie mniej zachodu, niż samo składanie znaków.

Program **POLODIAK** może znajdować się w dwóch zasadniczych stanach: wyłapywania i konwersji polskich liter oraz przepuszczania bajtów bez żadnej modyfikacji. Bieżący stan programu określa wartość bajtu statusowego **Status**. Wartość 0 oznacza wyłapywanie i konwersję, pozostałe wartości — transmisję bez konwersji. Transmisja bez konwersji dotyczy dwóch przypadków: sekwencji danych graficznych, gdy trzeba po prostu odliczyć podaną liczbę bajtów oraz analizę sekwencji sterującej, zapoczątkowanej znakiem ⟨ESCAPE⟩, w której niekiedy może być istotne zapamiętanie wartości niektórych bajtów (np. wskaźnika długości bloku graficznego po ⟨ESCAPE⟩ ⟨L⟩ itd.) lub rozpoznanie końca sekwencji argumentów (np. przy definiowaniu tabulatorów po ⟨ESCAPE⟩ ⟨D⟩).

Po rozpoznaniu znaku ⟨ESCAPE⟩, zwiastującego sekwencję sterującą, program „zapamiętuje sobie" ten fakt, ustawiając najstarszy bit bajtu statusowego. W związku z tym następny bajt będzie uznany za kod sekwencji. Program podejmie próbę odszukania kodu w tablicy kodów sterujących, zawierającej wszystkie akceptowane sekwencje 3- i więcej bajtowe. Jeśli kod w tablicy nie występuje to sekwencja zostanie uznana za dwubajtową i na tym jej obsługa się zakończy. Odnalezienie kodu w tablicy powoduje skok do odpowiedniego programu obsługi, który ogranicza się do ustawienia odpowiednich bitów sygnalizacyjnych w bajcie statusowym. Odpowiednio do stanu tych bitów będą traktowane kolejne bajty sekwencji sterującej. Tak np. po rozpoznaniu kodu: 'K', 'L', 'Y' lub 'Z', rozpoczynających blok graficzny, reakcja programu będzie następująca: zapamięta on w komórce **Licznik** dwa kolejne bajty, w których zakodowano długość bloku, a następnie przepuści do drukarki bez analizy taką liczbę bloku graficznego, jak podano. Dopóki trwa transmisja bloku graficznego, ustawiony jest najmłodszy bit bajtu statusowego, zakazujący dalszej analizy danych. Bit ten zostanie skasowany w chwili wyzerowania licznika.

Prawie wszystkie rozpatrywane sekwencje sterujące mają stałą długość — w naszym przypadku 3 lub 4 bajty (pomijamy najprostsze, dwubajtowe). Wyjątkiem jest sekwencja ⟨ESCAPE⟩ ⟨D⟩, służąca do definiowania w drukarce położeń tabulatorów. Długość sekwencji nie jest tu jednoznacznie określona, wiadomo natomiast, że ostatnim bajtem ma być 0. Sekwencja ta wymaga zatem specyficznego potraktowania. Program **POLODIAK** reaguje tylko na najczęściej spotykane sekwencje sterujące. W praktyce okazuje się to zupełnie wystarczające, ale jeśli zajdzie potrzeba, to nic nie stoi na przeszkodzie, aby uwzględnić i inne sekwencje. Tak np. drukarki standardu **EPSON** stosują pięciobajtową sekwencję inicjacji trybów graficznych: ⟨ESCAPE⟩ ⟨*⟩... Gdyby były potrzebne dodatkowe bity stanu, to do wykorzystania są jeszcze bity: 1 i 3 bajtu stanu. Można też oczywiście utworzyć nowe, dodatkowe komórki wskaźnikowe.

Aby uzyskać działający program za pomocą asemblera np. MASM, należy wprowadzić jego tekst źródłowy, zapisać w pliku tekstowym np. **POLODIAK.ASM**, po czym wydać zlecenie: **MASM POLODIAK**, kwitując kolejne pytania naciśnięciem klawisza ⟨ENTER⟩. Jeśli asemblacja zakończy się powodzeniem (0 błędów), przyjdzie pora na wywołanie konsolidatora (linkera): **LINK POLODIAK** i ⟨ENTER⟩ w odpowiedzi na kolejne pytania. Jeśli i konsolidator zakończy pracę poprawnie (meldunek o braku segmentu stosu jest normalny i nie należy się nim w tym przypadku przejmować), pozostaje przetworzyć program z postaci .EXE na .COM zleceniem: **EXE2BIN POLODIAK.EXE POLODIAK.BIN**. Wywoływanie programu odbywa się zleceniem: **POLODIAK** i powinno być wykonane tylko raz, najlepiej od razu po załadowaniu systemu operacyjnego.

Dla tych czytelników, którzy wprawdzie nie są biegli w posługiwaniu się asemblerem, ale mimo to pragnęliby skorzystać z powyższego programu, opracowaliśmy jak zwykle program ładujący w języku wysokiego poziomu. Tym razem posłużyliśmy się **TURBO-Pascalem**. Po wprowadzeniu programu do pamięci za pomocą edytora wystarczy program uruchomić. Jeżeli dane wprowadzono poprawnie, to na dysku domniemanym zostanie automatycznie utworzony plik **POLODIAK.COM** i na tym program pracę zakończy, w przeciwnym razie na ekranie pojawi się meldunek o błędzie.

Bardziej zaawansowana metoda nadrukowywania znaków mogłaby polegać na drukowaniu linii w dwóch przebiegach, bez cofania głowicy. W pierwszym przebiegu byłyby drukowane znaki podstawowe, w drugim — „nakładki", naturalnie tylko tam, gdzie byłoby to potrzebne. Oczywiście w tym przypadku niezbędne byłoby zorganizowanie w programie bufora, mieszczącego komplet danych składających się na jedną linię, włącznie ze wszystkimi znakami sterującymi, przełączającymi atrybuty pisma itd.

NOWOŚCI KOMPUTEROWE



SYSTEMS'87, czyli targi w „Krzemowej Bawarii"

GRZEGORZ ZALOT

Nie tak dawno zamieściliśmy na łamach „InforMika" relację z największych w Europie targów techniki komputerowej — CeBIT'87, odbywających się na wiosnę w Hanowerze. Tym razem zaprezentujemy krótką migawkę z innej imprezy o bardzo podobnym charakterze — targów SYSTEMS, organizowanych co dwa lata, jesienią w stolicy Bawarii — Monachium.

W ciągu ostatnich kilku lat bardzo dużo zainwestowano w przemysł elektroniczny rozmieszczony w tym rejonie Republiki Federalnej Niemiec. Mówi się nawet o „Silicon Bavaria" i „Isar Valley" — przez porównanie do kalifornijskiej Krzemowej Doliny — „Silicon Valley". Jako przykład niech posłuży fakt, że około 50% produkcji elementów elektronicznych w RFN zlokalizowane jest w okolicach Monachium.

Porównując SYSTEMS i CeBIT zauważa się bardzo duże podobieństwo (niemal ta [illegible] tematyka, bardzo podobny podział tematyczny).

Cała ekspozycja podzielona była na kilka grup. Osobno prezentowano oprogramowanie, elementy systemów komputerowych, urządzenia z zakresu telekomunikacji (transmisja danych itp.), systemy skomputeryzowanych miejsc pracy, wyposażenie stanowisk biurowych i centrów obliczeniowych.

Zapewne wielu czytelników oczekuje jakichś nowości w dziedzinie sprzętu zgodnego z IBM PC albo nowej rodziny PS/2. Tu spotka ich jednak zawód. Dlaczego? Otóż znaczna większość oprogramowania narzędziowego stosowanego w komputerach osobistych pracuje pod kontrolą systemu operacyjnego MS-DOS (różnymi jego wersjami), na dowolnej maszynie zgodnej w jakimś tam stopniu ze wzorcem IBM PC. Tymczasem pełne zalety większych modeli tej nowej klasy sprzętu (opisywanego już na naszych łamach) można wykorzystać tak naprawdę dopiero po zastosowaniu nowego systemu operacyjnego OS/2. Zalety tego systemu są bezsprzeczne, ale przy zastosowaniu oprogramowania wykorzystującego rozszerzony tryb procesorów 80286 i 80386. Obecnie programów wykorzystujących ten tryb jest bardzo niewiele, a na dobre przyjmą się one najwcześniej za jakieś dwa lata.

Praktycznie każdy komputer z procesorem 286 lub 386 może pracować pod kontrolą systemu, jeżeli tylko producent wygeneruje odpowiednią jego wersję. Na to jednak, aby system ten pracował rzeczywiście wydajnie, wymagana jest pamięć operacyjna rzędu 1,5 MB, a najlepiej jeszcze więcej (adresować można do 18 MB).

Dużym atutem rodziny PS/2 jest cena komputerów, bardzo korzystna przy wysokiej jakości produktu. Na horyzoncie jednak pojawiają się już [illegible] „czarne chmury" dla IBM — otóż jesienią ubiegłego roku dwie firmy: **Chips** i **Western Digital** opracowały specjalizowane układy wielkiej skali integracji realizujące w pełni funkcje sterowania mikrokanałami. Zestaw FE5400 firmy Western Digital wykonany w technice CMOS zapewnia pracę systemu przy częstotliwości rzędu do 20 MHz, przy czym jeden z układów zastępuje co najmniej kilkanaście stosowanych przez twórcę systemu (!). Dodatkowo zastąpienie płyty czterowarstwowej dwustronną umożliwi dodatkową redukcję kosztów o 33 procent. Na dzień



Compaq Deskpro 386 [illegible] — komputer o największej w swojej klasie zdolności obliczeniowej



Compaq Portable 386 — ta przenośna maszyna o [redacted] zaledwie 9 kg przewyższa swoją wydajnością niejeden znacznie większy komputer klasy IBM PC/AT

dzisiejszy: cena zestawu logiki [redacted]/2 wynosi tylko 99 dolarów...

Już dość dawno dała [redacted] tendencja do zwiększania mocy obliczeniowych mikrokomputerów osobistych. Trwa ona dalej, czego jednym z przykładów są nowe modele oferowane przez firmę Compaq. Najwydajniejszym (w czasie targów) modelem klasy PC był Compaq Deskpro 386/20 posiadający oczywiście procesor 80386. Jego moc obliczeniowa jest o ok. 25 procent większa niż w wypadku innych komputerów tej klasy z takim samym zegarem dzięki zastosowaniu tzw. Cache Memory — bardzo szybkiej pośredniej pamięci buforowej (pojemność 32 KB, czas dostępu 35 ns, pamięć statyczna), dopasowanej pod względem czasu dostępu do procesora. Pamięć ta jest obsługiwana przez specjalny sterownik Intel 82385. Możliwe jest zainstalowanie koprocesora arytmetycznego 80387 taktowanego z częstotliwością 20 MHz, co daje około czterokrotne przyspieszenie działań.

Bardzo dobrze już znany komputer przenośny Compaq Portable posiada również procesor 80386 [redacted] zegarem 20 MHz, a możliwe jest zainstalowanie koprocesora 80387 (20 MHz). Standardowa pamięć [redacted] pojemność 1 MB i czas dostępu 80 ns, co nieco spowalnia pracę mikroprocesora. Bardzo szybki jest dysk twardy dysponujący czasem dostępu poniżej 25 [redacted] przy 100 [redacted] pojemności.

Pozostańmy przy sprzęcie o najwyższej wydajności. Przykładem jest tu drukarka Seikosha SBP-10. Szybkość druku w trybie draft wynosi 800 znaków [redacted] sekundę, w trybie korespondencyjnym 100 zn/s, a w trybie wysokiej jakości 200 zn/s. Drukarka wyposażona jest [redacted] głowicę 18-igłową, przy czym igły ustawione są w dwóch równoległych rzędach, co umożliwiło uzyskanie bardzo wysokiej szybkości druku. Jeszcze wyższą jakość zapewniają drukarki z głowicą 24-igłową, [redacted] jednak okupione jest mniejszą szybkością pracy. Do takich należy Brother M-1724L. Szybkość druku

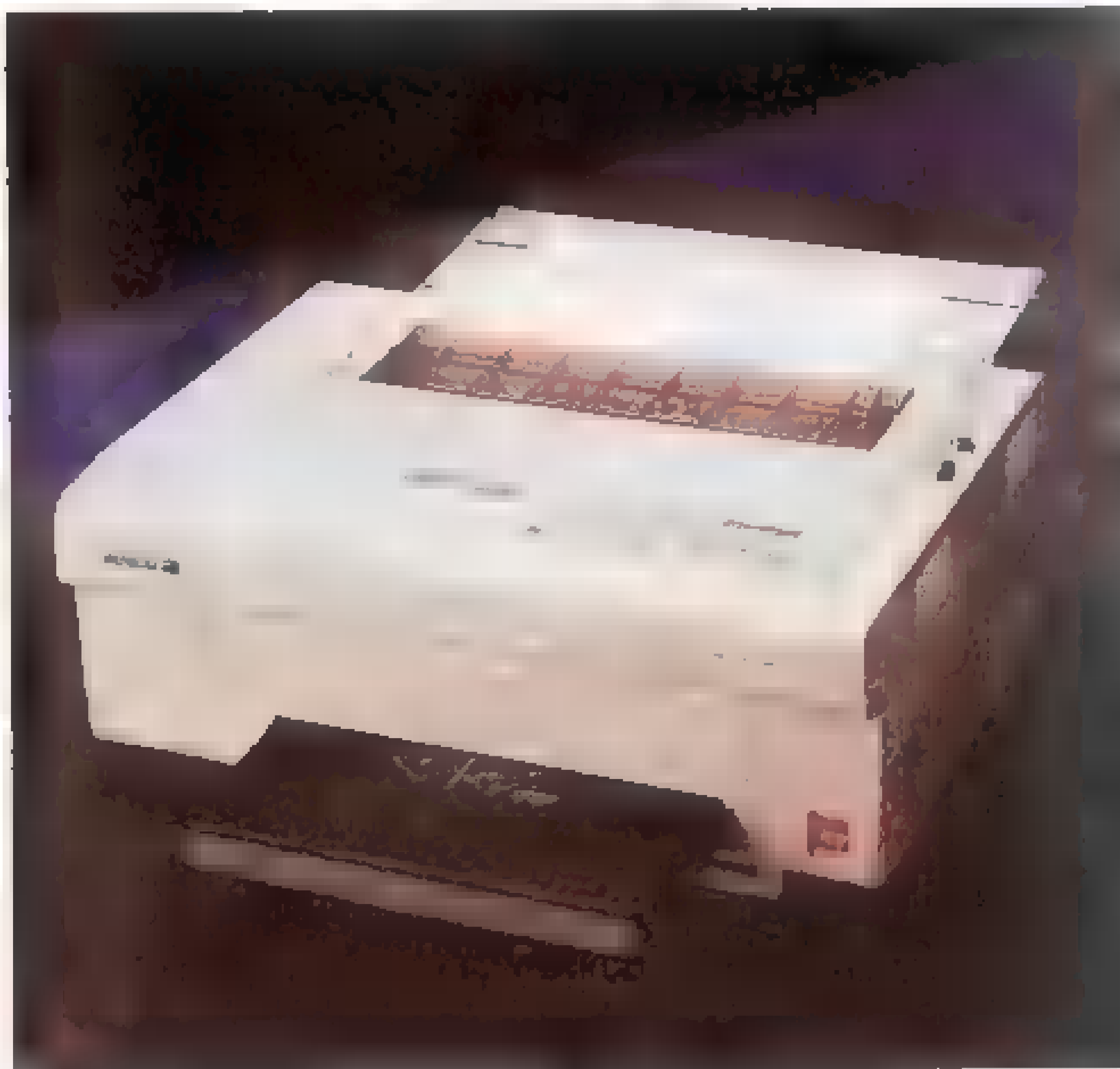


Drukarka Brother M-1724L — przy stosunkowo niedużych wymiarach [redacted] ona dość dużą szybkość druku wysokiej jakości dzięki zastosowaniu 24-igłowej głowicy



Nowy model Schneidera — PC 2640

Drukarka Seikosha SBP-10 — urządzenie o bardzo dużej wydajności. Szybkość druku 800 zn/s w trybie draft czyni ją najszybszą w swojej klasie



normalnego (lepszy od korespondencyjnego wielu drukarek) wynosi 216 zn/s, a druku o najwyższej jakości — 72 zn/s. Cenną [redacted] tej drukarki [redacted] niewielkie wymiary.

Każdy komputer musi być wyposażo-[redacted] w odpowiednią pamięć zewnętrzną, pozwalającą na przenoszenie danych. Danych tych [redacted] coraz więcej — wymagane są zatem większe pojemności pamięci zewnętrznych. Najpowszechniejszym rozwiązaniem stają się pamięci optyczne (opisane w relacji z CeBIT-u). Pewną konkurencją dla nich stają się zaprezentowane przez koncern Kodaka (firma Verbatim) jednostki dysków elastycznych 12 MB. Jednostka ta ma typową „połówkową" wysokość stacji do IBM PC i wykorzystuje dyskietki 5,25 cala umieszczone w sztywnej kopercie (podobnie jak dyskietki 3,5-calowe). Uzyskanie tak dużej pojemności możliwe stało się dzięki zastosowaniu technik stosowanych dotąd w dyskach twardych oraz dodatkowej mecha-[redacted] osłonie nośnika. Kilka danych technicznych: pojemność 12 MB (10 MB po sformatowaniu obustronnie), średni czas dostępu 65 [redacted], 64-bitowa korekcja błę-[redacted]w, szybkość transmisji 2,2 MB/s (wewnętrzna), żywotność nośnika 3,5 × 10 operacji na ścieżkę.

Inna propozycja Kodaka (a dokładnie firmy Verbatim), to system jednostek 5,25 cala o pojemności 3,3 lub 6,6 [redacted] (bez sformatowania). Zwiększenie pojemności uzyskano dzięki znacznie większej precyzji mechanizmu, nośnikiem jest typowa z wyglądu dyskietka, jednak o poprawionych parametrach warstwy magnetycznej.

Firma Schneider znana jest od pewnego czasu z produkcji komputerów w dużym stopniu zgodnych z IBM, a przy tym dość tanich. Ostatnio przybył jeszcze jeden model — [redacted]C 2640, zgodny z IBM PC/AT. Posiada on procesor 80286 z zegarem 12 MHz, standardowo kartę EGA, napęd 3,5 cala o pojemności 1,44 MB (jak IBM PS/2), Winchester 32 MB, RAM oczywiście 640 KB. Cena wersji podstawowej — około 4500 DM.

Nowoczesne komputery przenośne wyposażone są w wyświetlacze plazmowe (duży pobór mocy) lub ciekłokrystaliczne. W dziedzinie tej zauważa [redacted] dość duży postęp — przykładem może być wyświetlacz KL6440 firmy Data Modul. Przy wymiarach ekranu 217 × 138 mm zapewnia rozdzielczość 640 × 400 punktów (!), co jest zgodne z kilkoma standardami grafiki komputerów klasy IBM PC. Wyświetlacz taki posiada już na płytce [redacted]e specjalizowane układy zapewniające odpowiednie sterowanie [redacted]-ktrodami (współczynnik wypełnienia wynosi 1 : 200), natomiast organizacją ekranu zajmuje się zewnętrzny kontroler. [redacted] było się przekonać [redacted] własne oczy, jakość liter jest znakomita, nie obserwuje się migotania, [redacted]o taki typ wyświetlacza nie działa szkodliwie na wzrok operatora.

WORDSTAR a sprawa polska

TADEUSZ A. ZALESKI

JANUSZ J. MŁODZIANOWSKI

Jak wykazują badania od 60 do 70 procent czasu spędzanego z komputerem osobistym poświęcone jest przetwarzaniu tekstów (pisanie programu w dowolnym języku to również przetwarzanie tekstów!). Jednym z najpopularniejszych edytorów tekstowych (mimo „podeszłego wieku") jest obecnie produkt firmy Micro-Pro o nazwie WordStar. Popularny jest tak wśród użytkowników mikrokomputerów pracujących pod kontrolą systemu CP/M, jak i MS-DOS (PC-DOS). To jego popularność zdecydowała zapewne, że do kompilatorów języków Pascal, C i BASIC znanej firmy Borland International Inc. (Turbo-pascal, Turbo-C i Turbo-BASIC) dołączone są edytory wzorowane na edytorze WordStar, a można go również znaleźć w pakiecie SideKick tej samej firmy. Również edytor w pakiecie Norton Commander jest sterowany według reguł zapożyczonych z WordStara.

WordStar, dzięki wykorzystywaniu pracy monitora w trybie tekstowym, jest edytorem szybkim i sprawnym. Szereg dobrze pomyślanych opcji czyni z niego narzędzie, które dobrze „leży w ręku". Problemem, na który napotyka polski (i nie tylko!) użytkownik WordStara jest brak narodowych znaków diakrytycznych. Można sobie z tym poradzić na wiele sposobów — poniżej przedstawiamy jeden z nich.

Przed każdą „polską literą" należy umieścić specjalny znak, który będzie sygnalizował, że następująca po nim litera ma być interpretowana odmiennie, właśnie jako polski znak diakrytyczny. Do tego trzeba zdefiniować stosowne kształty tych znaków. Oczywiście musimy stworzyć procedurę, która będzie realizowała taką interpretację.

Wydruk 1

```
;                   PLFILTR            ed. 88/11/02     v. 1.10
CSEG                SEGMENT  'CODE
                    ASSUME   CS:CSEG   DS:CSEG

                    ORG      100H
Start:              JMP      Instal
;-----------------------------------------
```

WordStar wersja 4.00 — fragment tekstu z ukrytymi znakami kontrolnymi (polecenie ^OD)



```
Hdr         DB      0DH,0AH   ***********************
            DB      0DH,0AH   *      PL_FILTR       *
            DB      0DH,0AH   *                     *
            DB      0DH,0AH   *       v.1.10        *
            DB      0DH,0AH   *       (c)tez        *
            DB      0DH,0AH   ***********************
            DB      0DH,0AH   $
;----------------------------------
PrintPl:    CMP     AH,0      ;test, czy kod funkcji w AH = 0
                              wysłania jednego bajtu do drukarki
            JNE     Cr_k      gdy kod funkcji większy od 0 - powrót
            CMP     AL,1BH    czy wprowadzony znakiem jest <ESC>
            JNE     TsESC     ;jeśli nie - testuj dalej
            MOV     CS:CE     ustaw wskaźnik <ESC> w 1
            JMP     Koniec    ;wyślij <ESC> na drukarkę
;----------------------------------
TsESC:      CMP     CS:CE,1   czy poprzedni znak był <ESC>
            JNE     TsSwtch   jeśli nie  to testuj dalej
            CMP     AL,1EH    czy wprowadzonym znakiem jest <RS>
            JNE     TsESC1    jeśli nie  to wyślij go na drukarkę
            MOV     AL,54H    zamień na  T
            CMP     CS:SW,1   czy PL_Filtr jest aktywny
            JE      TsESC0    jeśli tak, to zablokować
            MOV     CS:SW,1   ;odblokuj PL_Filtr
            JMP     TsESC1
;----------------------------------
TsESC0      MOV     CS:SW,0   ;zablokuj PL_Filtr
TsESC1:     MOV     CS:CE,0   ustaw wskaźnik <ESC> w 0
            JMP     Koniec    wyślij analizowany znak na drukarkę
;----------------------------------
TsSwtch     CMP     CS:SW,1   czy PL_Filtr jest aktywny
            JNE     Cr_k      ;jeśli nie  to powrót
Cr_Q:       CMP     AL,1CH    ;czy wprowadzonym znakiem jest <FS>
                              ;WordStar zmienia ^Q na chr(1CH) = <FS>
            JNE     Cr_W      nie - testuj pozostałe możliwości
            MOV     CS:FS,1   ustaw wskaźnik znaku <FS> w 1
            JMP     Cr
;----------------------------------
Cr_W:       CMP     AL,1EH    ;czy wprowadzonym znakiem jest <RS>
                              ;WordStar zmienia ^W na chr(1EH) = <RS>
            JNE     TsCrQ     nie - testuj pozostałe możliwości
            MOV     CS:CW,1   ;ustaw wskaźnik znaku <RS> w 1
Cr:         MOV     AL,00H    na drukarkę wyślij <NUL>
Cr_k:       JMP     Koniec
;----------------------------------
TsCrQ:      CMP     CS:CQ,1   czy wskaźnik znaku <FS> jest w 1
            JNE     TsCrW     ;nie - testuj pozostałe możliwości
            MOV     CS:CQ,0   ustaw wskaźnik znaku <FS> w 0
            CMP     AL,61H    ;czy znak po <FS> to 'a'?
            JNE     TsQ1
            MOV     AL,60H    ;zamień na znak '`'
            JMP     Koniec
;----------------------------------
TsQ1:       CMP     AL,65H    'e'?
            JNE     TsQ2
            MOV     AL,7EH    '~'
            JMP     Koniec
;----------------------------------
TsQ2:       CMP     AL,65H    'e'?
            JNE     TsQ3
            MOV     AL,40H    '@'
            JMP     Koniec
;----------------------------------
TsQ3:       CMP     AL,6CH    'l'?
            JNE     TsQ4
            MOV     AL,7CH    '|'
            JMP     Koniec
;----------------------------------
TsQ4:       CMP     AL,6EH    'n'?
            JNE     TsQ5
            MOV     AL,5DH    ']'
            JMP     Koniec
;----------------------------------
TsQ5:       CMP     AL,6FH    'o'?
            JNE     TsQ6
            MOV     AL,7BH    '{'
            JMP     Koniec
;----------------------------------
TsQ6        CMP     AL,73H    's'?
            JNE     TsQ7
            MOV     AL,5EH    '^'
            JMP     Koniec
;----------------------------------
TsQ7:       CMP     AL,7AH    'z'?
            JNE     TsQ8
            MOV     AL,7DH    '}'
            JMP     Koniec
;----------------------------------
TsQ8:       CMP     AL,4CH    'L'?
            JNE     Koniec
            MOV     AL,5CH    '\'
TsQk:       JMP     Koniec
;----------------------------------
TsCrW:      CMP     CS:CW,1   czy wskaźnik znaku <RS> jest w 1
            JNE     TsSc      nie - analizuj dalej
            MOV     CS:CW,0
            CMP     AL,7AH    'z'?
            JNE     Koniec
            MOV     AL,5BH    '['
TsWk:       JMP     Koniec
;----------------------------------
```

```
TxtIc:      CMP     AL,40H   ;czy znakiem do druku jest '@'?
            JE      No_Pol   ;jeśli tak, to skocz do No_Pol
            CMP     AL,5BH   ;'['?
            JE      No_Pol
            CMP     AL,5CH   ;'\'?
            JE      No_Pol
            CMP     AL,5DH   ;']'?
            JE      No_Pol
            CMP     AL,5EH   ;'^'?
            JE      No_Pol
            CMP     AL,60H   ;'`'?
            JE      No_Pol
            CMP     AL,7BH   ;'{'?
            JE      No_Pol
            CMP     AL,7CH   ;'|'?
            JE      No_Pol
            CMP     AL,7DH   ;'}'?
            JE      No_Pol
            CMP     AL,7EH   ;'~'?
            JE      No_Pol
Koniec:     PUSHF            ;przygotuj stos
            CALL    CS:DWORD PTR Int17A
                             ;wywołaj obsługę drukarki w BIOS
            [redacted]
;-----------------------------------------
PrintAL:    MOV     AH,00H   ;wydrukuj znak o kodzie z AL
            PUSHF            ;przygotuj [redacted]
            CALL    CS:DWORD PTR Int17A
            RET              ;wywołaj obsługę drukarki w BIOS
;-----------------------------------------
Down_Load:  PUSH    AX       ;AL=0 - down_load_cancel
                             ;AL=1 - down_load_select
            MOV     AL,1BH   ;<ESC>
            CALL    PrintAL
            MOV     AL,25H   ;'%'
            CALL    PrintAL
            POP     AX       ;odtwórz zawartość AL
            CALL    PrintAL  ;wydrukuj
            MOV     AL,00H   ;wyślij chr(0) na drukarkę
            CALL    PrintAL
            RET
;-----------------------------------------
No_Pol:     PUSH    AX
            MOV     AL,00H   ;down_load_cancel
            CALL    Down_Load
            POP     AX       ;wydrukuj znak zdefiniowany
            CALL    PrintAL  ; w podstawowym zestawie znaków
            MOV     AL,01H   ;down_load_select
            CALL    Down_Load
            IRET
;-----------------------------------------
Int17A      DD      0        ;adres pierwotnej procedury w BIOS
CO          DW      0        ;wskaźnik znaku <FS>
CW          DW      0        ;wskaźnik znaku <RS>
CE          DW      0        ;wskaźnik znaku <ESC>
[redacted]  DW      1        ;wskaźnik aktywności
;-----------------------------------------
Instal:     LEA     DX,Hdr   ;adres nagłówka
            PUSH    DS       ;zachowaj na stosie DS
            MOV     AX,CS
            MOV     DS,AX    ;DS <- CS
            MOV     AH,09H   ;funkcja BIOS: wyślij łańcuch znaków
                             ; zakończonych znakiem '$'
            INT     21H      ;wołaj BIOS
            POP     DS       ;odtwórz DS
            MOV     AH,35H   ;funkcja DOS: podaj wektor przerwania
            [redacted] AL,17H ;numer przerwania obsługi drukarki
            INT     21H      ;wołaj DOS, powrót z wektorem w ES:BX
            MOV     WORD PTR Int17A,BX
            MOV     WORD PTR Int17A+2,ES
                             ;zapamiętaj pierwotny wektor przerwania
                             ;wskazujący procedurę w BIOS
            MOV     DX,OFFSET PrintPL
                             ;przesunięcie procedury PrintPL w DX
            MOV     AH,25H   ;funkcja DOS: ustaw wektor przerwania
            MOV     AL,17H   ;numer przerwania obsługi drukarki
            INT     21H      ;wołaj DOS, ustaw wektor dany w DS:DX
            MOV     DX,OFFSET Instal
                             ;przesunięcie = pierwszy wolny bajt
            INT     27H      ;uczyń procedurę PLFilte rezydującą
[redacted]  ENDS
            END     Start
```

Tablica 1 Kody ASCII przypisane polskim znakom diakrytycznym (por. BN-74/3101-01).

kod ASCII	znak ASCII	znak wg PN
40H	[redacted]	ę
5BH	[	ż
5CH	\	Ł
5DH	]	ń
5EH	^	ś
60H	`	ą
7BH	{	ó
7CH	|	ł
7DH	}	ź
7EH	~	ć

Tablica 2 Adresy pod którymi należy definiować znaki ^Q i ^W

WordStar, wersja	3.40	4.00
nazwa programu instalacyjnego	INSTALL	WSCHANGE
nazwa zmiennej definiującej ^Q	USR1	UPRQ
adres zmiennej definiującej ^Q	096Dh	087Eh
nazwa zmiennej definiującej ^W	USR2	UPRW
adres zmiennej definiującej ^W	097Ah	0897h

Wydruk 2

```
program PL_NLQ_GENERATOR;
const   vern='ver 1.20';                              {ed. 88/01/02}
        chsum=12470;
var sum: integer;
    disk:text;
    fname:string[12];

procedure header;
begin
  writeln;
  writeln('**************************************');
  writeln('*                                    *');
  writeln('*     PL_NLQ Characters Generator    *');
  writeln('*               '+vern+'             *');
  writeln('*                                    *');
  writeln('* Copyright (c) tez & jim 87/03/08  *');
  writeln('**************************************');
  writeln
end; {header}

procedure e(code:integer);
begin
   write(disk,chr(code)); sum:=sum+code
end; {e}

procedure initnl;
begin
    e( 27); e( 30);                                    {zablokuj PL_Filtr}
    e( 27); e(120); e(  1);                                   {select NLQ}
    e( 27); e( 58); e(  0); e(  0); e(  0);             {copy ROM to RAM}

    e( 27); e( 38); e(  0); e( 96); e( 96);            {kod  96, znak ą}
    e(  0);                                     {z przedłużeniem dolnym}
    e(  0); e(  0); e(  0); e(  8); e( 20); e( 64); e(  0); e( 64);
    e(  0); e( 64); e(  0); e( 64); e(  0); e( 64); e( 21); e(  2);
    e( 60); e(  0); e(  0); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e(  0); e( 24); e(  0); e(  0); e( 36); e(  0);
    e( 36); e(  0); e( 36); e(  0); e( 36); e(  0); e( 68); e(  1);
    e( 56); e(  5); e(  0); e(  5); e(  0); e(  0); e(  0);

    e( 27); e( 38); e(  0); e(126); e(126);            {kod 126, znak ć}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e( 12); e( 16); e(  0); e(  2); e(  0); e( 32);
    e(  0); e( 32); e(  0); e( 32); e(  0); e( 96); e(  0); e(  2);
    e(  0); e(176); e(  0); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e( 24); e(  4); e(  0); e( 32); e(  0); e(  2);
    e(  0); e(  2); e(  0); e( 66); e(  0); e(  2); e(  0); e(160);
    e(  0); e( 56); e(  0); e(  0); e(  0); e(  0); e(  0);

    e( 27); e( 38); e(  0); e( 64); e( 64);            {kod  64, znak ę}
    e(  0);                                     {z przedłużeniem dolnym}
    e(  0); e(  0); e( 24); e( 32); e( 16); e(  4); e( 16); e( 64);
    e( 16); e( 64); e( 16); e( 64); e( 17); e( 66); e( 16); e(  4);
    e( 16); e( 32); e( 16); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e( 48); e(  8); e(  0); e( 64); e(  0); e(  4);
    e(  0); e(  4); e(  0); e(  4); e(  2); e(  5); e(  0); e( 65);
    e(  0); e(  9); e( 32); e(  0); e(  0); e(  0); e(  0);
    e( 27); e( 38); e(  0); e(124); e(124);            {kod 124, znak ł}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e(  0); e(  0); e(  0); e(  0); e(  0); e(  0);
    e(128); e(  0); e(136); e(  0); e(254); e(  0); e( 16); e(  0);
    e( 16); e(  0); e(  0); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e(  0); e(  0); e(  0); e(  0); e(  0); e(  0);
    e( 10); e(  0); e(  2); e(  0); e(254); e(  0); e(  2); e(  0);
    e( 34); e(  0); e(  0); e(  0); e(  0); e(  0); e(  0);

    e( 27); e( 38); e(  0); e( 93); e( 93);            {kod  93, znak ń}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e( 32); e(  0); e( 62); e(  0); e(  0); e(  0);
    e( 32); e(  0); e( 32); e(  0); e( 32); e( 64); e( 32); e(  0);
    e( 32); e(128); e( 30); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e(  2); e(  0); e( 62); e(  0); e( 34); e(  0);
    e(  0); e(  0); e(  0); e( 64); e(  0); e(  0); e(  0); e(128);
    e(  2); e( 32); e( 30); e(  0); e(  2); e(  0); e(  0);

    e( 27); e( 38); e(  0); e(123); e(123);            {kod 123, znak ó}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e( 12); e( 16); e(  0); e(  2); e(  0); e( 32);
    e(  0); e( 32); e(  0); e( 32); e(  0); e( 96); e(  0); e(  2);
    e(  0); e(144); e( 12); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e( 34); e(  4); e(  0); e( 32); e(  0); e(  2);
    e(  0); e(  2); e(  0); e( 66); e(  0); e(  2); e(  0); e(160);
    e(  0); e(  4); e( 24); e(  0); e(  0); e(  0); e(  0);

    e( 27); e( 38); e(  0); e( 94); e( 94);            {kod  94, znak ś}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e( 18); e(  0); e( 10); e( 32); e(  0); e( 32);
    e(  8); e( 32); e(  0); e( 32); e(  8); e( 96); e(  8); e( 32);
    e(  0); e(130); e( 32); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e( 22); e( 32); e(  0); e(  2); e(  0); e(  2);
    e(  0); e(  2); e(  0); e( 66); e(  0); e(  2); e(  0); e(130);
    e( 40); e(  0); e( 36); e(  0); e(  0); e(  0); e(  0);

    e( 27); e( 38); e(  0); e(125); e(125);            {kod 125, znak ź}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e( 48); e(  2); e( 32); e(  0); e( 32); e(  4);
    e( 32); e(128); e( 32); e(138); e( 32); e(  0); e( 32); e( 16);
    e( 32); e(  0); e( 34); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e( 34); e(  0); e(  2); e(  4); e(  2); e(  0);
    e(  2); e(136); e(  2); e(128); e(  2); e( 16); e(  2); e(  0);
    e(  2); e( 32); e(  6); e(  0); e(  0); e(  0); e(  0);

    e( 27); e( 38); e(  0); e( 91); e( 91);            {kod  91, znak ż}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e( 48); e(  3); e( 33); e(  0); e( 32); e(  4);
    e( 32); e(  0); e( 32); e(  8); e( 96); e(  0); e( 32); e( 16);
    e(160); e(  0); e( 34); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e( 34); e(  0); e(  2); e(  4); e(  2); e(  0);
    e(  3); e(  8); e( 66); e(  0); e(  2); e( 16); e(130); e(  0);
    e(  2); e( 32); e(  6); e(  0); e(  0); e(  0); e(  0);

    e( 27); e( 38); e(  0); e( 92); e( 92);            {kod  92, znak Ł}
    e(128);                                   {bez przedłużenia dolnego}
    e(  0); e(  0); e(128); e(  0); e(136); e(  0); e(254); e(  0);
    e(144); e(  0); e(128); e(  0); e(  0); e(  0); e(  0); e(  0);
    e(  0); e(  0); e(  0); e(  0); e(  6); e(  0); e(  0);
    e(  0); e(  0); e( 10); e(  0); e(  2); e(  0); e(254); e(  0);
    e(  2); e(  0); e( 34); e(  0); e(  2); e(  0); e(  2); e(  0);
    e(  2); e(  0); e(  2); e(  0); e( 14); e(  0); e(  0);
```

```
    s[ 27], s[ 37]; s[ [1, s[ 0]          {select download chr set}
    s[ 27]; s[ 30]                              {uaktywnij PL_Filtr}
end; {Instal}

begin
    header;
    sum:=0;
    namein:='PLNLQGEN.CHR';
    assign(disk,namein);
    rewrite(disk);
    stuln;
    if sum<>chsum then
    begin
       writeln('Niezgodność sumy kontrolnej!! ');
       rewrite(disk)
    end    {sum<>chsum}
    else
       writeln('nazwa zbioru wynikowego: ',namein);
    close(disk)
end.
```

Wydruk 3

```
a     copy plnlqgen.chr lpt1

b     copy con plnlq.bat
      copy plnlqgen.chr lpt1
      ^Z
```

Zacznijmy od końca. Procedura PL_FILTR (por. wydruk 1) zostaje [redacted] chwili wywołania (etykieta Instal) zainstalowana [redacted] pamięci komputera jako procedura rezydująca, stając się częścią programu obsługującego drukarkę (przerwanie 17H). Ilekroć przerwanie to zostanie wywołane, spowoduje uaktywnienie procedury. Dokładny opis tego mechanizmu znajdzie czytelnik [redacted] artykule R. Wacławka pt. „Jak IBM PC obsługuje klawiaturę?" („InforMik" 2/87, str. 9).

Jako znaki poprzedzające polskie litery przyjęliśmy znak ‹[redacted]› (kodASCII = 1CH) dla ą, ć, ę, ł, [redacted], ó, ś, ź i Ł oraz znak ‹RS› (kodASCII = 1EH) dla litery ż. Procedura po rozpoznaniu znaku ‹FS› lub ‹RS› (etykiety Cr_Q i Cr_W), ustawia odpowiedni wskaźnik (CQ) lub (CW), nadając mu wartość 1, zaś do drukarki wysłany zostaje znak NUL (kodASCII = 00H). Jeżeli kolejny znak należy do zestawu interesujących nas znaków a, c, e, ..., to w jego miejsce (etykiety TsCrQ i TsCrW) zostanie wysłany kod znaku (zgodnie z polską normą BN-74/3101-01, por. tabela 1), pod którym w pamięci RAM drukarki znajduje się stosowny kształt polskiej litery (o tym dokładniej poniżej). Równocześnie wskaźnikowi (CQ) lub (CW) zostaje przypisana wartość 0.

Jeżeli procedura wykryje, że do drukarki skierowany został znak o kodzie zajętym przez jedną z polskich liter (etykieta TsSc), [redacted] zostanie on wydrukowany za pomocą procedury No_Pol. Przed wysłaniem analizowanego znaku

WordStar wersja 4.00 — fragment tekstu z widocznymi znakami sygnalizującymi obecność polskich znaków diakrytycznych

zostaje przywołany w drukarce oryginalny kształt znaków znajdujący się w pamięci ROM (down_load_cancel), następuje druk znaku i ponowne przywołanie zestawu znaków zdefiniowanego przez użytkownika (down_load_select).

I byłoby to wszystko, gdyby nie fakt, [redacted] możliwość sterowania pracą drukarki (zmiana sposobu drukowania, indeksy, ustawienie marginesów, itd.) za pomocą znaków kontrolnych rozpoczynających się od znaku ‹ESC› (kodASCII = 1BH). Po nim jako argument [redacted] wystąpić znak, który jest przedmiotem „szczególnego zainter[redacted]" omawianej procedury. By uniknąć takich kolizji wprowadzony został trzeci wskaźnik (CE): nadana mu zostanie wartość 1, gdy analizowany będzie znak ‹ESC›. Wówczas kolejny znak (etykieta TsESC) zostanie wyprowadzony [redacted] drukarkę bez jakiejkolwiek modyfikacji. Jest tylko jeden wyjątek: ciąg znaków ‹ESC› ‹RS› nie jest stosowany jako kod sterujący drukarki. Wykrycie takiego ciągu znaków powoduje wyłączenie ponowne — włączenie aktywności filtra (etykiety TsESC i TsESCO). Informacja o stanie aktywności procedury jest przechowywana [redacted] zmiennej o nazwie SW. Możliwość wyłączania filtra okazała się niezbędna [redacted] względu na procedurę definiowania znaków (wydruk 2).

Kształty liter w drukarce typu Star NL-10 definiuje się przez wysłanie do niej odpowiedniego ciągu znaków sterujących. By ułatwić czytelnikowi zrozumienie całego procesu, załączamy program w języku Pascal, który generuje zbiór PLNLQGEN.CHR zawierający zestaw znaków sterujących dla drukarki NL-10. Za jego pomocą definiujemy w trybie NLQ (*Near Letter Quality*) kształty polskich liter zestawionych w tabeli 1. Po zablokowaniu procedury rezydującej PL_FILTR i wybraniu trybu NLQ, kopiuje się oryginalne kształty liter z pamięci ROM drukarki do RAM. Następnie dla każdej litery należy podać jej kod, położenie i ciąg 46 (2* 23) kodów definiujących kształt dla każdego z dwóch przebiegów głowicy drukarki. Na końcu, jako obowiązujący wybrany zostaje przedefiniowany [redacted] znaków i uaktywniona zostaje procedura PL_FILTR. Zbiór PLNLQGEN.CHR zostanie wygenerowany, gdy program stwierdzi poprawność sumy kontrolnej (sum = chsum). Inicjacja drukarki polega teraz na skopiowaniu zbioru PLNLQGEN.CHR do drukarki (por. wydruk 3a). Polecenie to może być treścią zbioru typu BATCH (wydruk 3b), wtedy inicjację drukarki otrzyma się przez proste zlecenie PLNLQ.

Ostatnim krokiem jest zadbanie o to, by w tekstach generowanych przez edytor WordStar przed polskimi znakami umieszczane były kody znaków ‹FS› i ‹RS›. Należy to uczynić za pomocą znaków kontrolnych edytora definiowanych przez użytkownika ^Q i ^W (‹CTRL — Q› i ‹CTRL — W›). Autorzy edytora przewidzieli cztery znaki kontrolne (^Q, ^W, ^E i ^R), które użytkownik może samodzielnie zdefiniować stosownie do swoich potrzeb. Definicje należy umieścić w kodzie edytora pod wskazanymi (por. tabela 2) adresami: dla ^Q „01H — 1CH" i dla ^W „01H — 1EH". Można to uczynić za pomocą programu instalującego edytor.

Zaznaczenie polskiego znaku przy pracy z tak dostosowanym edytorem odbywa się przez wprowadzenie ^PQa dla ą, ^PQe dla ę, ..., ^PQz dla ź i ^PWz dla ż. Przy dłuższych polskich tekstach jest to trochę uciążliwe. Należy więc przedefiniować klawiaturę. Najprościej można tego dokonać za pomocą programu rezydującego SuperKey (Borland International Inc.). Definiujemy klawisz ‹ALT›a jako ^PQa, ‹ALT›e jako ^PQe itd. Naciśnięcie teraz ‹ALT›a spowoduje wygenerowanie pożądanego ciągu ^PQa. Można oczywiście skonstruować... drugi filtr, który będzie kontrolował pracę klawiatury.

Uważny czytelnik z pewnością zauważył, że procedura PL_FILTR [redacted] ze zbiorem PLNLQGEN.CHR pozwala na wydruk dowolnego zbioru tekstu z uwzględnieniem polskich liter. Jedynym warunkiem jest sygnalizowanie obecności polskich znaków przez poprzedzanie ich znakami ‹FS› i ‹RS›. Jest to istotna [redacted] proponowanego rozwiązania.



DYSK PAMIĘCIOWY w ZX SPECTRUM

CZĘŚĆ II

DARIUSZ ADAM PRZYGODA KRZYSZTOF AMBORSKI

W pierwszej części artykułu zamieszczonego [redacted] poprzednim numerze „Infor-Mika" podany został opis sprzętowej części rozszerzenia pamięci komputera ZX Spectrum. Obecnie zajmiemy się programem obsługi tak utworzonego dysku pamięciowego.

Program obsługi dysku pamięciowego napisany został [redacted] języku asemblera mikroprocesora Z80. Zajmuje 1 [redacted] pamięci począwszy od adresu #3900.

Program powoduje rozszerzenie interpretera BASIC-a ZX Spectrum o pięć dodatkowych instrukcji, których formaty wzorowane [redacted] na oryginalnych instrukcjach mikrokomputera. Sposób realizacji tego rozszerzenia bazuje na przechwytywaniu obsługi błędów składniowych. Sposób [redacted] – pomimo swojej elegancji syntaktycznej – charakteryzuje się występowaniem nieuniknionych niedogodności związanych z architekturą systemu ZX Spectrum. Przechwytywanie obsługi błędu polega na [redacted]nie zawartości zmiennej systemowej (ERR SP) wskazującej [redacted] adres procedury obsługi błędu. Wynika stąd konieczność reinicjalizacji programu po każdej akcji systemu, w efekcie której zawartość tej zmiennej jest odnawiana. Sytuacja taka występuje (oprócz koniecznej inicjalizacji po wczytaniu programu do pamięci) w przypadku użycia instrukcji NEW, CLEAR i RUN. Dokładniejsze omówienie problemu przechwytywania obsługi błędów można znaleźć w artykule M. Goreckiego pt. „Jak rozszerzyć BASIC ZX Spectrum" zamieszczonym w miesięczniku „Informatyka" 6/85 i 7/85.

Inicjalizacja dokonywana jest za pomocą dowolnej instrukcji o argumencie **USR 15000** (najbezpieczniej **RANDOMIZE**).

Podczas odwoływania się do obszaru dysku pamięciowego program zakłada jego uporządkowanie. Dlatego też przed pierwszym odwołaniem po instalacji programu bezwzględnie należy uporządkować obszar dysku instrukcją **FORMAT***. Niewykonanie powyższego może [redacted] nieprzewidziane efekty z załamaniem [redacted] systemu włącznie. W praktyce [redacted] jest to taka duża niedogodność — wystarczy w procedurze ładującej program obsługi dysku umieścić instrukcję **FORMAT***.

W stosunku do analogicznych procedur BASIC-a dotyczących obsługi magnetofonu program nakłada na użytkownika drastyczne ograniczenia. Spowodowane są one szczupłością [redacted]ejsca przewidzianą na program i założenie niestosowania nakładek obszarów pamięciowych. Ograniczenia te [redacted]: konieczność operacji jedynie na blokach bajtów (bloki typu **CODE**) i nieakceptowanie wartości grup parametrów podanych niejawnie (np. rozszerzenia **SCREEN$**). Otwiera się tu pole do popisu dla zdolnych programistów — kosztem obszaru dysku można wykonać nakładki będące procedurami obsługi poszczególnych instrukcji i ściągać je do pamięci aktywnej na czas wykonywania instrukcji rozszerzając w ten sposób możliwości programu.

Szczupłość obszaru pamięciowego podyktowała również sposób przechowywania plików w obszarze dysku. Nie występuje tu pojęcie sektora i tablicy alokacji — pliki umieszczone [redacted] w pamięci sekwencyjnie (jeden za drugim) w postaci (nagłówek pliku 1 — plik 1 — nagłówek pliku 2 — plik 2 — ...). Bezpośrednio po ostatnim pliku w następną komórkę pamięci wpisywany jest bajt #FF sygnalizujący koniec obszaru zajętego przez dane. Ten nieefektywny sposób gospodarki zasobami powoduje długi czas realizacji instrukcji **ERASE*** — do kilkunastu [redacted]kund, co [redacted] również pośrednią przyczyną reakcji programu na próbę zapisania pliku o już istniejącej nazwie (program wraca z raportem błędu).

Nagłówek pliku ma długość 14 bajtów i następującą strukturę: bajty 0—9 stanowią [redacted] pliku (uzupełnioną spacjami), bajty 10—11 zawierają długość pliku, bajty 12—13 zawierają adres początkowy bloku.

Program do swojej pracy wykorzystuje piętnaście bajtów pamięci **RAM**, gdzie lokuje swoje zmienne. Są to: umieszczone kolejno bezpośrednio za obszarem programu — dziesięciobajtowy bufor **NAMBUF** i dwubajtowe **LENBUF** i **STRBUF** (stanowią one odwzorowanie nagłówka) oraz lokacja o adresie 23681 (nie używana lokacja w obszarze zmiennych systemo-

```
[redacted] *L-
20         ORG   14592
30  INIT0  LD    HL,(23613)
[redacted]        LD    (HL),INPUT0?256
[redacted]        INC   HL
60         LD    (HL),INPUT0/256
70         RET
80
90  INPUT0 LD    A,(23610)
100        CP    #0B
110        JR    Z,NONSNS
120        CP    #17
130        JR    Z,NONSNS
140 ERROR  CALL  [redacted]
150        JR    NZ,RUNERR
160 SYNERR LD    HL,INPUT$
170        PUSH  HL
180        JP    #1287
190 RUNERR CALL  #1303
200        LD    (IY),#FF
210        LD    HL,(23641)
220        CALL  #11A7
230        LD    HL,INPUT$
240        PUSH  HL
250        JP    #12B4
260
270 RETED  LD    HL,[redacted]
280        PUSH  HL
290        LD    HL,#1287
300 RETED1 PUSH  HL
310        JP    #1B76
320 RETRUN LD    HL,INPUT$
330        JR    RETED1
340
350 NONSNS RST   #18
360        CP    "*"
370        JR    NZ,ERROR
380        DEC   HL
390        LD    A,(HL)
400        INC   HL
410        INC   HL
420        CP    #FB
430        JR    Z,SAVE
440        CP    #EF
450        JP    Z,LOAD
460        CP    [redacted]
470        JP    Z,ERASE
480        CP    #CF
490        JP    Z,CAT
500        CP    #D0
510        JP    Z,FORMAT
520        JR    [redacted]
530
540 XPTR   LD    (IY),#FF
550        LD    (IY+38),[redacted]
560        RET
570
580 EOLIN  RST   #18
590        CP    ":"
600        RET   Z
610        CP    [redacted]
620        RET   Z
630        SCF
640        RET
650
660 NAME   CALL  #2BF1
670        LD    A,B
680        OR    C
690        RET   Z
700        LD    A,#F6
710        ADD   A,C
720        JR    NC,NAME1
730        LD    C,10
740 NAME1  LD    B,0
750        EX    DE,HL
760        LD    DE,NAMBUF
770        LDIR
780        SCF
790        RET
800
810 PRPBUF LD    HL,NAMBUF
820        LD    BC,#0E20
830 PRP1   LD    (HL),C
840        INC   HL
850        DJNZ  PRP1
860        RET
870
880 SAVE   CALL  PRPBUF
890        RST   [redacted]
900        CALL  XPTR
910        LD    HL,R1SAVE
920        PUSH  HL
930        CALL  #1C8C
940 R1SAVE BIT   7,(IY)
950        JP    Z,ERROR
960        POP   HL
970        CALL  [redacted]
980        [redacted]    Z,SAVE1
990        CALL  NAME
1000       LD    (IY),#0E
1010       JP    NC,ERROR
1020       CALL  FNDSTR
1030       JP    C,ERROR
1040 SAVE1 RST   #18
1050       CP    #AF
1060       JP    NZ,CODADR
1070       RST   #20
1080       CALL  XPTR
1090       LD    HL,R2SAVE
1100       PUSH  HL
1110       CALL  #1C7A
1120 R2SAVE BIT  7,(IY)
1130       JP    Z,ERROR
1140       POP   HL
1150       CALL  #2530
1160       JR    Z,SAVE2
1170       LD    (IY),25
1180       CALL  #2DA2
1190       JP    NZ,ERROR
1200       JP    C,ERROR
1210       LD    (LENBUF),BC
1220       CALL  #2DA2
1230       JP    NZ,ERROR
1240       JP    C,ERROR
1250       LD    (STRBUF),BC
1260       CALL  XPTR
1270 SAVE2 CALL  EDLIN
1280       JP    C,ERROR
1290 SAVEXE CALL #2530
1300       JP    Z,RETED
1310       CALL  [redacted]
1320       PUSH  HL
1330       LD    DE,(LENBUF)
1340       LD    B,A
1350       SCF
1360       [redacted]   HL,DE
1370       ADC   A,0
1380       POP   HL
1390       CP    [redacted]
1400       JR    C,SAVE5
1410       LD    (IY),3
1420       JP    ERROR
1430 SAVE5 LD    A,B
1440       LD    BC,14
1450       LD    IX,NAMBUF
1460       CALL  LDBNK
1470       LD    BC,[redacted]
1480       LD    IX,(STRBUF)
1490 SAVE6 CALL  LDBNK1
1500       EX    AF,AF'
1510       OUT   (#FF),A
1520       LD    (HL),#FF
1530       XOR   A
1540       [redacted]   (#FF),A
1550       JP    RETRUN
1560
1570 FNDNXT PUSH AF
1580       PUSH  HL
1590       LD    BC,10
1600       [redacted]   HL,BC
1610       CALL  GETBNK
1620       OUT   (#FF),A
1630       LD    E,(HL)
1640       LD    BC,1
1650       [redacted]   HL,BC
1660       JR    NC,FND1
1670       INC   A
1680       OUT   (#FF),A
1690       SET   7,H
1700 FND1  LD    D,(HL)
1710       XOR   [redacted]
1720       OUT   (#FF),A
1730       POP   HL
1740       POP   AF
1750       LD    BC,14
1760       ADD   HL,BC
1770       ADC   A,0
1780       ADD   HL,DE
1790       ADC   A,0
1800       RET
1810
1820 FNDEND XOR  A
1830       LD    HL,32768
1840 FNDEN1 CALL ASKEND
1850       RET   NZ
1860       CALL  FNDNXT
1870       JR    FNDEN1
1880
1890 ASKEND PUSH AF
1900       [redacted]  HL
1910       CALL  GETBNK
1920       OUT   (#FF),A
1930       LD    B,(HL)
1940       [redacted]
1950       OUT   (#FF),A
1960       POP   HL
1970       POP   AF
1980       BIT   7,B
1990       RET
2000
2010 GETBNK SLA  H
2020       RLA
2030       SRL   H
2040       SET   7,H
2050       RET
2060
2070 X_ENT JP    INIT0 ;*ENTER[redacted]
2080
2090 NXTBNK EX   AF,AF'
2100       INC   A
2110       EX    AF,AF'
2120       LD    HL,32768
2130       RET
2140
2150 LDBNK CALL  GETBNK
2160       EX    AF,AF'
2170 LDBNK1 LD   D,(IX)
2180       EX    AF,AF'
2190       OUT   (#FF),A
2200       EX    AF,AF'
2210       LD    (HL),D
2220       XOR   [redacted]
2230       OUT   (#FF),A
2240       CALL  LOOPCT
2250       JR    NZ,LDBNK1
2260       RET
2270
2280 LOOPCT INC  IX
2290       INC   HL
2300       DEC   BC
2310       LD    A,H
2320       OR    L
2330       CALL  Z,NXTBNK
2340       LD    A,B
2350       OR    C
2360       RET
2370
2380 LDMEM CALL  GETBNK
2390       EX    AF,AF'
2400 LDMEM1 EX   AF,AF'
2410       OUT   (#FF),A
2420       EX    AF,AF'
2430       LD    D,(HL)
2440       XOR   [redacted]
2450       OUT   (#FF),A
2460       LD    (IX),D
2470       CALL  LOOPCT
2480       JR    NZ,LDMEM1
2490       RET
2500
2510 CMPSTR LD   IX,NAMBUF
2520       LD    B,10
2530       LD    DE,1
2540       OUT   (#FF),A
2550       EX    AF,AF'
```

```
2560 CMPST2 LD   A,(HL)
2570        CP   (IX)
2580        JR   NZ,NOTEQ
2590        [redacted] AF,AF'
2600        INC  IX
2610        ADD  HL,DE
2620        JR   NC,CMPST1
2630        INC  A
2640        OUT  (#FF),A
2650        SET  7,H
2660 CMPST1 EX   AF,AF'
2670        DJNZ [redacted]
2680 NOTEQ  XOR  A
2690        OUT  (#FF),A
2700        CP   B
2710        RET
2720
2730 FNDSTR LD   HL,32768
2740        XOR  A
2750 FNDST1 CALL ASKEND
2760        JR   NZ,GOTEND
2770        PUSH AF
2780        PUSH HL
2790        CALL GETBNK
2800        CALL CMPSTR
2810        POP  HL
2820        JR   NC,GOTSTR
2830        POP  AF
2840        CALL FNDNXT
2850        JR   FNDST1
2860 [redacted] AND [redacted]
2870        RET
2880 GOTSTR POP  AF
2890        SCF
2900        RET
2910
2920 [redacted] CALL FNDSTR
2930        RET  NC
2940        LD   IX,NAMBUF
2950        LD   BC,14
2960        CALL LDMEM
2970        SCF
2980        [redacted]
2990
3000 [redacted] CALL PRPBUF
3010        RST  #20
3020        CALL XPTR
3030        LD   HL,R1LOAD
3040        PUSH HL
3050        CALL #1C8C
3060 R1LOAD BIT  7,(IY)
3070        JP   Z,ERROR
3080        POP  HL
3090        [redacted] #AF
3100        JR   NZ,CODADR
3110        RST  #20
3120        CALL EOLIN
3130        JR   C,CODADR
3140        CALL #2530
3150        JP   Z,RETED
3160        CALL LOADXX
3170        JP   NC,ERROR
3180        LD   IX,(STRBUF)
3190 LDCONT LD   BC,(LENBUF)
3200        CALL LDMEM1
3210 LDEND  CALL XPTR
3220        JP   RETRUN
3230
3240 LOADXX LD   (IY),#0E
3250        CALL NAME
3260        CALL GETHDR
3270        RET  NC
3280        DEC  B
3290        RET  [redacted]
3300        SCF
3310        RET
3320
3330 CODADR LD   HL,R2LOAD
3340        PUSH HL
3350        CALL #1C82
3360 R2LOAD BIT  7,(IY)
3370        JP   Z,ERROR
3380        POP  HL
3390        CALL EOLIN
3400        JP   C,ERROR
3410        CALL #2530
3420        JP   Z,RETED
3430        LD   (IY),25
3440        CALL [redacted]
3450        JP   NZ,ERROR
3460        JP   C,ERROR
3470        PUSH [redacted]
3480        CALL LOADXX
3490        POP  IX
3500        JP   NC,ERROR
3510        JR   LDCONT
3520
3530 DELETE CALL FNDSTR
3540        PUSH [redacted]
3550        PUSH AF
3560        CALL GETBNK
3570        LD   DE,1
3580        EX   AF,AF'
3590        EXX
3600        POP  AF
3610        EX   (SP),HL
3620        CALL FNDNXT
3630        CALL GETBNK
3640        LD   DE,1
3650        LD   IX,23681
3660 DEL3   OUT  (#FF),A
3670        LD   B,(HL)
3680        LD   (IX),B
3690        EX   AF,AF'
3700        EXX
3710        OUT  (#FF),A
3720        LD   B,(IX)
3730        LD   (HL),B
3740        LD   B,2
3750 DEL2   LD   (IX),B
3760        ADD  HL,DE
3770        JR   NC,DEL1
3780        SET  7,H
3790        INC  A
3800        CP   [redacted]
3810        JR   Z,DELEND
[redacted] DEL1 EX AF,AF'
3830        EXX
3840        LD   B,(IX)
3850        DJNZ DEL2
3860        EX   AF,AF'
3870        EXX
3880        JR   DEL3
3890 DELEND OUT  (#FF),A
3900        POP  HL
3910        EXX
3920        RET
3930
3940 ERASE  CALL PRPBUF
3950        RST  #20
3960        CALL XPTR
3970        LD   HL,R1ERAS
3980        PUSH HL
3990        CALL #1C8C
4000 R1ERAS BIT  7,(IY)
4010        JP   Z,ERROR
4020        POP  [redacted]
4030        CP   #AF
4040        JP   C,CODADR
4050        RST  #20
4060        CALL EOLIN
4070        JP   C,CODADR
4080        CALL #2530
4090        JP   Z,RETED
4100        CALL NAME
4110        LD   (IY),#[redacted]
4120        CALL FNDSTR
4130        JP   NC,ERROR
4140        CALL DELETE
4150        CALL XPTR
4160        JP   RETRUN
4170
4180 CAT    RST  #20
4190        CALL EOLIN
4200        JP   C,ERROR
4210        CALL XPTR
4220        CALL #2530
4230        JP   Z,RETED
4240        LD   A,2
4250        CALL #1601
4260        LD   DE,#9ED
4270        LD   BC,5
4280        CALL #203C
4290        LD   B,7
4300        CALL PRBSP
4310        LD   DE,#9A2
4320        LD   BC,5
4330        CALL #203C
4340        LD   B,3
4350        CALL PRBSP
4360        LD   BC,7
4370        LD   DE,TEXT1
4380        CALL #203C
4390        LD   HL,32768
4400        XOR  [redacted]
4410 CAT1   CALL ASKEND
4420        JP   NZ,RETRUN
4430        LD   IX,NAMBUF
4440        LD   BC,14
4450        PUSH [redacted]
4460        PUSH HL
4470        CALL LDMEM
4480        LD   DE,NAMBUF
4490        LD   BC,10
4500        CALL #203C
4510        LD   A," "
4520        RST  #10
4530        LD   A," "
4540        RST  #10
4550        LD   BC,(STRBUF)
4560        CALL #2D2B
4570        CALL #2DE3
4580        LD   A,#17
4590        RST  #10
4600        LD   A,20
4610        RST  #10
4620        LD   A,0
4630        RST  #10
4640        LD   BC,(LENBUF)
4650        CALL #2D2B
4660        CALL #2DE3
4670        LD   A,#0D
4680        RST  #10
4690        POP  HL
4700        POP  AF
4710        CALL FNDNXT
4720        JR   CAT1
4730
4740 PRBSP  PUSH BC
4750        LD   A," "
4760        RST  #10
4770        POP  BC
4780        DJNZ PRBSP
4790        RET
4800
4810 FORMAT RST  #20
4820        CALL EOLIN
4830        JP   C,ERROR
4840        CALL XPTR
4850        CALL #2530
4860        JP   Z,RETED
4870 FORM2  LD   A,1
4880        OUT  (#FF),A
4890        LD   A,#FF
4900        LD   (32768),A
4910        XOR  A
4920        OUT  (#FF),A
4930        JP   RETRUN
4940
4950 TEXT1  DEFM "Lenght"
4960        DEFB #0D
4970
4980 NAMBUF DEFS 10
4990 LENBUF DEFS 2
5000 STRBUF DEFS 2
5010
5020        DEFB #7F
5030        DEFM " D.A.P. '87"
5040
5050 *D+
5060 *L+
5070        END
```

wych). Umieszczenie jej w pamięci 16 KB miało na celu przyspieszenie działania instrukcji **ERASE***.

Program został podany w postaci źródłowej i przystosowany do asemblacji za pomocą popularnego asemblera GENS 3. Ponieważ niejednokrotnie korzysta on z procedur zawartych w ROM ZX Spectrum, celowe wydaje się zapoznanie się przed przystąpieniem do jego analizy z artykułem [illegible] Kaczmarczuka „Komputery Sinclaira — rachunki numeryczne w kodzie maszynowym" („Informatyka" 6/85), jak również z pozycją I. Logana i F. O'Hary „The Complete Spectrum ROM Dissembly" (dostępna na giełdach w postaci kserokopii).

Poniżej zamieszczony został skrócony opis procedur programu, mający na celu ułatwienie ich analizy. Występujące w nim pojęcie X__YZ oznacza liczbę trzybajtową B1-B2-B3 umieszczoną w rejestrze X i parze rejestrów YZ.

Wyjaśnienia wymaga także problem organizacji procedur bezpośredniej obsługi banków pamięci; ponieważ założeniem przy pisaniu programu było użycie stosu maszynowego komputera w miejscu założonym przez system, więc wszystkie procedury komunikacji z bankami pamięci o numerze różnym od zera musiały być realizowane bez użycia stosu.



Opis zestawu dodatkowych instrukcji dostarczanych przez program

SAVE* *nazwa* **CODE** *start*, *długość* — instrukcja powoduje przepisanie do obszaru dysku pamięciowego bloku bajtów z pamięci jako pliku o nazwie *nazwa*. Parametry *start* i *długość* mają znaczenie identyczne jak w instrukcji **SAVE**. Sposób podania parametrów może być jawny (jako teksty i liczby, np. [illegible], 12,44) lub niejawny (jako zmienne, np. a,b). Nazwa musi być niepusta i różna od wszystkich nazw plików już zapisanych w obszarze dysku pamięciowego, w przypadku niespełnienia tych warunków nastąpi powrót do systemu z raportem „Invalid file name". Parametry liczbowe muszą być nieujemne i mniejsze od 2^{16}, w przeciwnym przypadku nastąpi powrót do systemu z raportem „Parameter error". W przypadku próby zapisu pliku dłuższego niż ilość wolnego miejsca w obszarze dysku pamięciowego nastąpi powrót do systemu z raportem „Out of memory".

LOAD* *nazwa* **CODE** *start* — instrukcja powoduje przepisanie do pamięci bloku bajtów z obszaru dysku pamięciowego zapisanych tam pod nazwą *nazwa*. Jeżeli podany był adres *start* blok zapisywany jest do, począwszy od tego adresu, w przeciwnym razie od adresu zawartego w nagłówku pliku. Ograniczenia parametrów są identyczne jak przy instrukcji **SAVE***. Próba wywołania instrukcji z nazwą pustą lub nie istniejącą kończy się powrotem do systemu z raportem „Invalid file name".

ERASE* *nazwa* **CODE** — instrukcja powoduje usunięcie z obszaru dysku pamięciowego pliku o nazwie *nazwa*. Ograniczenia parametru *nazwa* są takie same, jak w instrukcji **LOAD***.

CAT* — instrukcja powoduje skatalogowanie zawartości dysku pamięciowego. Wyświetlane są nazwy, długości i adresy startowe wszystkich plików aktualnie zapisanych w obszarze dysku.

FORMAT* — instrukcja powoduje wpisanie do pierwszej komórki dysku pamięciowego wartości FF; powoduje to pozorne wymazanie wszystkich zapisanych już plików. Instrukcja nie narusza plików fizycznie (za wyjątkiem pierwszej komórki nazwy pierwszego pliku).

Opis procedur programu obsługi dysku pamięciowego

INIT Procedura ustawiająca adres obsługi błędów na wartość adresu etykiety **INPUT**.

INPUT Procedura obsługi błędów o kodach #0B (komunikat „Nonsense in Basic") i 17 („Invalid stream"); są to błędy występujące przy analizie składni instrukcji używanych przez program obsługi dysku. Zawiera ona także realizację powrotów do systemu ZX Spectrum w trybach wykonywania programu i analizy składniowej.

NONSNS Procedura wstępnej analizy składniowej linii programu zawierającej instrukcję obsługi dysku — po rozpoznaniu kodu instrukcji dokonuje ona skoku do programu jej obsługi.

XPTR Procedura ustawiająca początkowe wartości zmiennych systemowych **ERR NR** i **XPTR**.

EOLIN Procedura sprawdzająca poprawność składniową końca linii; w wypadku błędu zwraca ustawiony znacznik CY.

NAME Procedura sprawdzająca parametry zmiennej łańcuchowej opisanej deskryptorem znajdującym się na szczycie stosu kalkulatora i przeładowująca ją do obszaru bufora programu obsługi dysku (od adresu **NAMBUF**).

PRPBUF Procedura zapełniająca obszar bufora programu obsługi dysku (od adresu **NAMBUF**) spacjami.

SAVE Procedura obsługi instrukcji **SAVE***; realizuje analizę składniową i wykonanie instrukcji.

FNDNXT Procedura znajdująca adres początku następnego pliku w obszarze dysku. Wymaga podania adresu bieżącego pliku w postaci A__HL, zwraca adres początku następnego pliku w tej samej postaci. Nie sprawdza poprawności danych wejściowych — zakłada, że następny plik musi istnieć.

FNDEND Procedura szukająca końca obszaru zajętego przez dane w obszarze dysku. Zwraca adres pierwszej wolnej lokacji w postaci A__HL.

ASKEND Procedura testująca, czy dana lokacja w obszarze dysku jest pierwszą wolną lokacją. Wymaga podania testowanej lokacji w postaci numeru banku w A i adresu bezwzględnego w HL. W przypadku sukcesu zwraca wyzerowany znacznik Z.

GETBNK Procedura przetwarzająca format adresu lokacji w obszarze dysku. Wymaga podania adresu lokacji w postaci A__HL,

zwraca ten sam adres w postaci numeru banku [redacted] A i adresu bezwzględnego w HL.

X_ENT Adres wejściowy procedury (15 000).

NXTBNK Procedura ustawiająca adres następnego banku. Wymaga podania numeru bieżącego banku [redacted] A. Zwraca numer następnego banku [redacted] A i adres bezwzględny = 8000 [redacted] HL. Nie sprawdza poprawności numeru banku.

LDBNK Procedura zapisu bloku [redacted]nych z pamięci do obszaru dysku. Przenosi BC bajtów z pamięci począwszy od adresu zawartego w IX do obszaru banku, począwszy od adresu danego w postaci A_HL. Nie sprawdza poprawności danych wejściowych.

LDMEM Procedura odczytu bloku danych z obszaru dysku do pa[redacted]. Przenosi [redacted] bajtów z obszaru dysku począwszy [redacted] adresu zawartego w IX do obszaru banku począwszy od adresu danego [redacted] postaci A_HL. Nie sprawdza poprawności danych wejściowych.

CMPSTR Procedura porównująca zmienne łańcuchowe zawarte [redacted] buforze programu obsługi dysku (od adresu NAMBUF) i pierwszych dziesięciu komórkach obszaru dysku, począwszy od adresu bezwzględnego danego [redacted] HL [redacted] banku o numerze danym [redacted] B. W przypadku równości łańcuchów zwraca wyzerowany znacznik CY i [redacted]rową zawartość rejestru B.

FNDSTR Procedura szukająca adresu zmiennej łańcuchowej równej wzorcowi. Wymaga podania wzorca w buforze programu obsługi dysku (od adresu NAMBUF). W przypadku sukcesu zwraca ustawiony znacznik CY i adres pierwszej komórki (zajętej przez zmienną w obszarze dysku) w postaci A_HL. W przypadku nieznalezienia zmiennej zwraca wyzerowany znacznik CY.

GETHDR Procedura odzyskująca parametry pliku o zadanej nazwie. Wymaga podania nazwy pliku [redacted] obszarze bufora programu obsługi dysku (od adresu NAMBUF). [redacted] przypadku znalezienia pliku zwraca jego parametry [redacted] obszarze bufora programu obsługi dysku (długość od adresu LENBUF, adres startu od adresu LENBUF) oraz ustawiony znacznik CY. W przypadku nieznalezienia pliku zwraca wyzerowany znacznik CY.

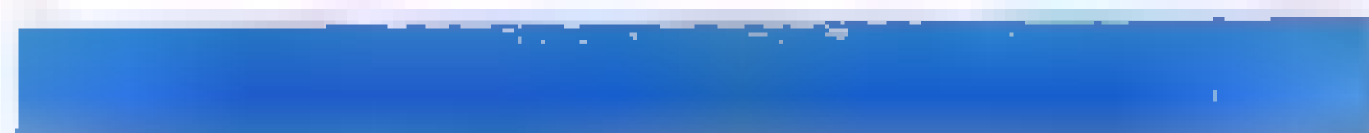
LOAD Procedura obsługi instrukcji LOAD*; realizuje analizę składniową i wykonanie instrukcji.

DELETE Procedura usuwająca plik [redacted] obszaru dysku. Wymaga podania nazwy pliku do usunięcia w obszarze bufora programu obsługi dysku (od adresu NAMBUF). Nie sprawdza poprawności danych wejściowych (plik o danej nazwie musi istnieć!).

ERASE Procedura obsługi instrukcji ERASE*; realizuje analizę składniową i wykonanie instrukcji.

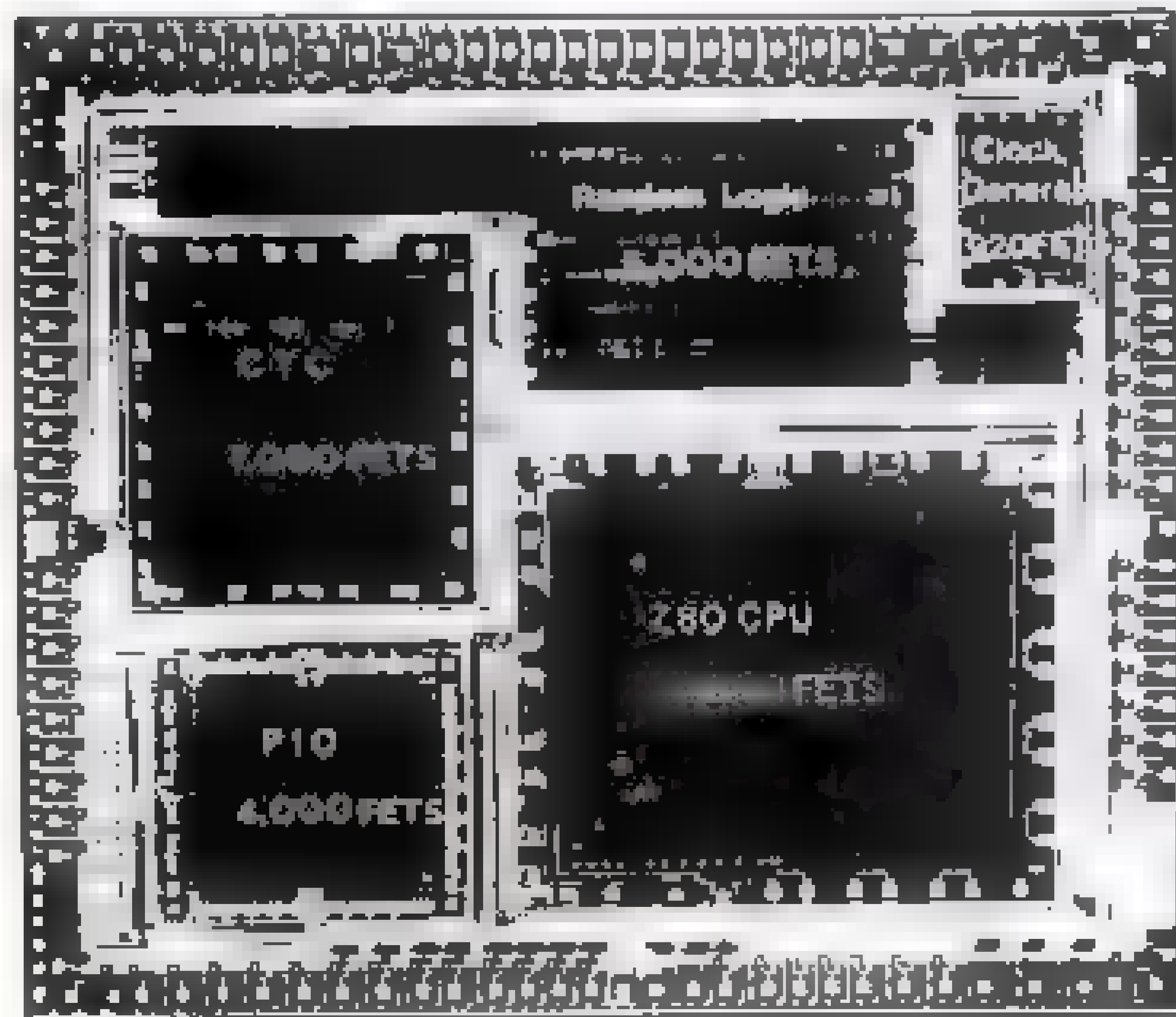
CAT Procedura obsługi instrukcji CAT*; realizuje analizę składniową i wykonanie instrukcji.

FORMAT Procedura obsługi instrukcji FORMAT*; realizuje analizę składniową i wykonanie instrukcji.



Rozwój technologii produkcji układów scalonych spowodował gwałtowny spadek [redacted] układów wielkiej i bardzo wielkiej skali integracji (LSI i VLSI). Konsekwencją spadku [redacted] było [redacted] powszechniejsze stosowanie w miejsce mikroprocesorów ośmiobitowych mikroprocesorów o długości słowa 16 i [redacted] bity, gdyż zyski wynikające z ich większych możliwości przetwarzania danych skutecznie rekompensowały poniesione nakłady [redacted]. Istnieją jednak urządzenia, w których stosowanie mikroprocesorów o długości słowa większej niż [redacted] bitów jest nieuzasadnione (w większości są to podsystemy peryferyjne).

W połowie lat siedemdziesiątych firma Intel, wychodząc [redacted] przeciw potrzebom konstruktorów sprzętu mikroprocesorowego, opra[redacted]ła rodzinę mikrokomputerów jednoukładowych serii [redacted], z których każdy stanowił mikroprocesor zintegrowany na jednym płatku krzemu z pamięcią i układami specjalizowanymi. Układy te znalazły wprawdzie szerokie zastosowanie [redacted] konstrukcjach urządzeń peryferyjnych, [redacted] nieoficjalnym standardem, [redacted] którym oparły się konstrukcje tego typu urządzeń, stał się mikroprocesor Zilog [redacted] i układy specjalizowane jego rodziny. Powodem tego są duże możliwości przetwarzania tego mikroprocesora.



Niesłabnąca popularność [redacted] powodowała powstawanie coraz nowszych (szybszych i oszczędniejszych energetycznie) jego wersji. Ostatnie słowo [redacted] tej dziedzinie należy do firmy Toshiba, która w 1987 r. ogłosiła wprowadzenie [redacted] handlu jednostrukturowych komputerów rodziny Z80.

Niecodzienność pomysłu polega na wykorzystaniu opracowań istniejących już struktur. Standardowe struktury stanowiące mikroprocesor [redacted] CPU) generator przebiegu zegarowego (clock), port równoległy (PIO) i układ czasowy (CTC) zostały po prostu umieszczone obok siebie na jednym płatku krze[redacted] (odtworzono nawet pola kontaktowe każdej ze struktur). W ostatniej fazie procesu produkcji techniką metalizacji [redacted] połączenia pomiędzy poszczególnymi blokami składowymi i gotowa płytka jest konfekcjonowana. Obecnie dostępne [redacted] trzy różniące się konfiguracją wersje systemu pakowane w obudowy o 84 lub 100 końcówkach. Zastosowana technologia CMOS zapewnia pracę przy częstotliwości przebiegu zegarowego 4, 6 [redacted] 8 MHz.

Zaletami tego rozwiązania (w stosunku [redacted] analogicznej [redacted]strukcji złożonej z oddzielnych układów scalonych) [redacted]: niższa [redacted] i większa niezawodność. Ponieważ nakłady poniesione na wyko[redacted] procesu metalizacji końcowej są stosunkowo niewielkie, [redacted] możliwość konfigurowania jednostrukturowego systemu zgodnie z życzeniem zamawiającego, który optymalność doboru konfiguracji mógł sprawdzić, testując wersję makietową złożoną z oddzielnych kostek.

Firma Toshiba zapowiada kontynuację prac i zwiększenie asortymentu dostępnych konfiguracji przy jednoczesnym ciągłym ich doskonaleniu.

(dap)

Szybkie wyszukiwanie INFORMACJI

Tadeusz [redacted]

Tekst poniższy przeznaczony jest specjalnie dla dość licznej grupy zwolenni[redacted] „szykanowanego" ostatnio poczciwego BASIC-a, którzy zupełnie słusznie uważają, że do rozwiązania wielu praktycznych problemów jest on narzędziem zupełnie wystarczającym. Z drugiej strony miłośnicy asemblera będą mogli podpatrzeć jak [redacted] im zapewne z BASIC-a (z którego zdążyli już „wyrosnąć") konstrukcje w prosty sposób przenieść do asemblera.

Pisane [redacted] BASIC-u programy typu baz danych oprócz wielu zalet mają jedną wadę. W kilku krytycznych punktach najbardziej nawet elegancko (czytaj: własnoręcznie) opracowanego menu, dość skutecznie „zatykają się" [redacted] wprowadzeniu odpowiednio dużej liczby danych. Jednym z takich punktów jest wyszukiwanie rekordu [redacted] pliku. Wydłużający się czas trwania tej operacji doprowadza autora takiego programu do wniosku że: widać tak być musi, bo program nie jest „maszynowy". Jest to na szczęście tylko częściowa prawda, bo zamiast przerabiać cały program wystarczy najczęściej dorobić małą „protezkę" w asemblerze, która radykalnie uzdrowi sytuację. Ponadto programującego [redacted] BASIC-u może w ogóle [redacted] interesować jak ją zrobiono. Dla własnej wygody takie usprawnienie może uważać za dodatkową instrukcję BASIC-a. Wychodząc [redacted] tego założenia napisano już kilka wersji „Beta BASIC-a". Niestety [redacted] zawsze możemy pozwolić sobie na luksus zajęcia 10 do 20 KB pamięci operacyjnej, tym bardziej, [redacted] te[redacted] typu programach (bazy danych) liczy się każdy wolny bajt. Za proponowanym przez nas rozwiązaniem przemawia fakt, że proponowana przez autora każda taka „protezka" nie będzie zajmowała więcej niż 200—300 bajtów pamięci operacyjnej. Poza tym, zawsze do programu będziemy mogli dołączyć tylko [redacted], które są nam naprawdę potrzebne.

Sformułowanie problemu, czyli o [redacted] nam właściwie chodzi?

Załóżmy, że dysponujemy w programie bazy danych pewną tablicą znakową dwuwymiarową. Kolejne wiersze tej tablicy zapełniają rekordy o stałej długości równej drugiemu wymiarowi tablicy. Identyfikację odpowiednich pól rekordu zapewnia nam operacja wycinania z danego wiersza tablicy łańcucha składającego się ze znaków od zadanej kolumny początkowej do [redacted]danej kolumny końcowej. Naszym zadaniem jest znalezienie numeru wiersza tablicy, w którym [redacted] zadanym polu znajduje się poszukiwany łańcuch. [redacted]esztkę wątpliwości powinien ostatecznie rozwiać poniższy wydruk algorytmu, do którego prowadzi przedstawione uprzednio rozumowanie.

WYDRUK 1

```
10 REM POSZUKIWANIE LANCUCHA A$
11 REM W TABLICY [illegible] B$
12 REM OD WIERSZA M DO WIERSZA L
13 REM OD KOLUMNY K-TEJ
14 REM [illegible] ZNALEZIONO
20 REM [illegible]
30 LET [illegible]
40 FOR [illegible] TO L
50   FOR [illegible]
60   IF [illegible] THEN GO TO 90
70   NEXT [illegible]
80   LET [illegible]
90 NEXT [illegible]
100
110 RETURN
```

Zauważmy dodatkowo, że przyjęliśmy tu możliwość przeszukiwania niekoniecznie całej tablicy. Głównym powodem tego ulepszenia jest fakt, że niewiele komplikuje on rozwiązanie, a może się jeszcze do czegoś przydać. Teraz przystępujemy do zakodowania procedury [redacted] asemblerze.

Opis procedury w asemblerze

Procedura składa [redacted] [redacted] dwu odrębnych podprogramów: przekazania parametrów wywołania i podprogramu właściwego przeszuk[redacted] tablicy. Do przekazania parametrów z BASIC-a wykorzystano funkcję definiowaną. W momencie wywołania funkcji interpreter umieszcza w zmiennej systemowej DEFADD (adres zmiennej 23563) adres aktualnych parametrów wywołania. Adres ten jest [redacted] na pierwszym znaku za nawiasem otwierającym [redacted] definicji funkcji (linia 10 programu demonstracyjnego [redacted] BASIC-u). Dla interpretera jest obojętne, w którym miejscu programu zostanie umieszczona definicja funkcji, jednak ze względu na liniowe przeszukiwanie obszaru programu lepiej umieszczać definicje [redacted] pierwszych liniach (przemawia za tym także czytelność programu). W procedurze maszynowej wymieniony adres jest wpisywany [redacted] rejestru indeksowego IX, [redacted] następnie pobierane są kolejne parametry. Istotna jest tu różnica we wstawianiu w miejsce parametrów formalnych parametrów aktualnych typu liczbowego i typu łańcuchowego. Parametry liczbowe [redacted] umieszczane bezpośrednio w miejsce formalnych, natomiast w drugim przypadku są wpisywane adresy, pod którymi znajdują się łańcuchy. Następnie jest wywoływany podprogram przeszukiwania, który jest prawie dokładną kopią algorytmu napisanego w BASIC-u. Jedynym istotnym dodatkiem jest fragment od linii 930 do linii 1040. Zostaje [redacted] nim obliczony adres kolejnego wiersza przeszukiwanej tablicy. Jest to o tyle konieczne, [redacted] struktura tablicy rozpoznawana przez interpreter BASIC-a na podstawie indeksów nie jest oczywista z poziomu asemblera. Wykorzystano tu procedurę zawartą

w ROM, mnożącą dwie liczby zawarte [illegible] rejestrach DE i HL (wynik mnożenia w HL). Ponadto dla większej przejrzystości zachowano na ile to możliwe nazwy zmiennych z algorytmu w BASIC-u. Jak widać [illegible] końcu wydruku programu źródłowego numer wiersza tablicy, w którym znaleziono poszukiwany łańcuch jest wpisany do pary rejestrów BC. Z tego powodu będzie on po powrocie do BASIC-a równy wartości funkcji USR. Jeśli łańcuch nie zostanie znaleziony, funkcja USR przyjmie wartość zero, co nie spowoduje

WYDRUK 2

```
10 ;PROCEDURA  INARRAY
20 ;  ZX SPECTRUM 48k
30 ;(C) T.Basista 1986
40 ;-------------------------
50                   ORG  64770
60 DEFADD            EQU  23563
70 MULTHD            EQU  #2AF4
80                   JR   START
90 ;-------------------------
100 ;Zmienne
110 ;adres 1 elem.tabl d$
120 ;pierwszy wym.tabl d$
130 ;drugi wymiar.tabl d$
140 ;adres aktual.wiersza d$
150 ;-------------------------
160 DS1              DEFS 2
170 LV               DEFS 2
180 RV               DEFS 2
190 DSI              DEFS 2
200 ;-------------------------
210 ;adres pierw.elem.x$
220 ;dlugosc lancucha x$
230 XS1              DEFS 2
240 XSL              DEFS 2
250 ;-------------------------
260 ;-------------------------
270 ;nr wiersza pocz.szukania
280 ;nr kolumny pocz.szukania
290 ;nr wiersza aktual.porow.
300 ;nr wiersza znalezionego
310 ;nr kolumny aktual.porow.
320 VM               DEFS 2
330 VN               DEFS 2
340 VI               DEFS 2
350 VK               DEFS 2
360 VJ               DEFS 2
370
380 ;-------------------------
390 START            DI
400                  CALL PARAME
410                  CALL SZUKAJ
420                  EI
430                  RET
440 ;-------------------------
450 PARAME           LD   IX,(DEFADD)
460
470 ;adres ds1
480                  LD   L,(IX+4)
490                  LD   H,(IX+5)
500                  LD   (DS1),HL
510
520 ;drugi wymiar d$
530                  DEC  HL
540                  LD   D,(HL)
550                  DEC  HL
560                  LD   E,(HL)
570                  LD   (RV),DE
580
590 ;pierwszy wymiar d$
600                  DEC  HL
610                  LD   D,(HL)
620                  DEC  HL
630                  LD   E,(HL)
640                  LD   (LV),DE
650
660 ;adres xs1
670                  LD   L,(IX+13)
680                  LD   H,(IX+14)
690                  LD   (XS1),HL
700
710 ;dlugosc x$
720                  DEC  HL
730                  LD   D,(HL)
740                  DEC  HL
750                  LD   E,(HL)
[illegible]          LD   (XSL),DE
770
780 ;poczat. wiersz szukania
790                  LD   L,(IX+22)
800                  LD   H,(IX+23)
810                  [illegible] (VM),HL
820
830 ;poczat kolumna szukania
840                  LD   L,(IX+30)
850                  LD   H,(IX+31)
860                  LD   (VN),HL
870                  RET
880 ;-------------------------
890 ;let k=0
900 SZUKAJ           LD   HL,0
910                  LD   (VK),HL
920
930 ;adres d$(m)
940                  LD   HL,(VM)
950                  DEC  HL
960                  LD   DE,(RV)
[illegible]          CALL MULTHD
980                  EX   DE,HL
990                  LD   HL,(DS1)
1000                 ADD  HL,DE
1010                 LD   DE,(VN)
1020                 ADD  HL,DE
1030                 DEC  HL
1040                 LD   (DSI),HL
1050
1060 ;for i=m to 1
[illegible]          LD   HL,(VM)
1080                 [illegible] (VI),HL
1090
1100 L40             [illegible] HL,(LV)
1110                 LD   DE,(VI)
1120                 XOR  A
1130                 SBC  HL,DE
1140                 JR   C,L110
1150
1160 ;for j=1 to lenx$
1170                 LD   HL,1
1180                 LD   (VJ),HL
1190
1200 L50             LD   HL,(XSL)
1210                 LD   DE,(VJ)
1220                 XOR  A
1230                 SBC  HL,DE
1240                 JR   C,L80
1250
1260 ;if d$(i)(j)<>x$(j) then
1270 ;go to 90
1280                 LD   HL,(VJ)
1290                 DEC  HL
1300                 PUSH HL
1310                 LD   DE,(DSI)
1320                 ADD  HL,DE
1330                 LD   A,(HL)
1340                 POP  HL
1350                 LD   DE,(XS1)
1360                 AD   HL,DE
1370                 CP   (HL)
1380                 JR   NZ,L90
1390
1400 ;next j
1410                 LD   HL,(VJ)
1420                 INC  HL
1430                 LD   (VJ),HL
1440                 JR   L50
1450
1460 ;let k=i:go to 110
1470 L80
1480                 [illegible] HL,(VI)
1490                 LD   (VK),HL
1500                 JR   L110
1510
1520 ;adres pola w nast.rek.
1530 L90             LD   HL,(DSI)
1540                 LD   DE,(RV)
1550                 ADD  HL,DE
1560                 LD   (DSI),HL
[illegible] ;next i
1580                 LD   HL,(VI)
1590                 INC  HL
1600                 LD   (VI),HL
1610                 JR   L40
1620
1630 ;return
1640 L110            LD   BC,(VK)
1650                 RET
1660 ;-------------------------
1670 ;koniec listingu
```

niejednoznaczności, gdyż w wersji BASIC-a na Spectrum tablice mogą być indeksowane wyłącznie od wartości większych od zera (wydruk 3).

Jak wykorzystać opisaną procedurę praktycznie...

Przedstawioną procedurę zastosowano praktycznie w programie umożliwiającym założenie podręcznego słownika (np. polskiego i angielskiego). Zauważmy tu, że nie jest istotne, [illegible] którego języka, na który zamierzamy tłumaczyć. Dostęp do danych jest tak szybki, [illegible] ustalanie kolejności alfabetycznej wyrazów nie jest konieczne. Wynika stąd, że nie są potrzebne dwie wersje przyporządkowań wyrazów, co jest warunkiem koniecznym stosowania klasycznych słowników. Fakt ten umożliwia zredukowanie do połowy wielkości pamięci operacyjnej przeznaczonej do wprowadzenia informacji. Zauważmy, że w przypadku trzech języków zysk jest czterokrotny. Oczywiście prezentowanemu rozwiązaniu można zarzucić zbyt małą pojemność (1000 par wyrazów 12-literowych). Można jednak zauważyć, że jest to wystarczająca ilość, by zniechęcić do kolejnego przepisywania strona za stroną słownika do komputera. Znacznie ciekawsza będzie sytuacja, w której czytelnik znający w stopniu elementarnym język angielski przystąpi do tłumaczenia technicznego tekstu o tematyce, która go pasjonuje (czytaj: czasopisma komputerowe). Wtedy, o ile ma dobrą pamięć, w miarę zbliżania się do końca tekstu będzie coraz rzadziej kartkował słownik. W przeciwnym przypadku, coraz częściej będzie się denerwować, tym że „szukał dopiero przed chwilą tego wyrazu i szkoda, że go gdzieś nie zapisał". Dysponując przedstawionym poniżej programem najpierw nieznane słowo wystuka na klawiaturze i albo od razu się dowie co ono oznacza, albo dopisze dodatkowo tłumaczenie, co nie jest zajęciem tak nudnym jak bezmyślne przepisywanie. Reasumując: Uczmy się języków obcych!... a póki co dla tych, którzy nie znają asemblera drukujemy jak zwykle program ładujący procedurę maszynową.

WYDRUK 3

```
1 REM PROGRAM DEMONSTRACYJNY PROCEDURY "INARRAY"
2 REM "ZX SPECTRUM 48K"
3 REM (C) Tadeusz Basista 1986
9
10 DEF FN I(D$,X$,M,N)=USR 64770
20 CLEAR 64769: LOAD "INARRAY MC"CODE 64770,227
30 DIM D$(1000,24): POKE 23609,10: POKE 23658,8
99
100 REM MENU
110 GO SUB 220: PRINT "MENU SLOWNIKA"
120 PRINT "1-TLUMACZENIE": PRINT "2-REDAGOWANIE"
130 PRINT "3-ODCZYT": PRINT "4-ZAPIS"
140 GO SUB 910
150 IF KL<1 OR KL>4 THEN GO TO 100
160 GO SUB 210: GO SUB KL*1000: GO TO 100
199
200 REM PROCEDURY POMOCNICZE
210 PRINT #2;AT KL,0; BRIGHT 1;Z$: RETURN
220 BEEP .3,20: CLS : RETURN
230 INPUT "SLOWO:"; LINE X$: RETURN
240 INPUT "NAZWA "; LINE N$: RETURN
250 PRINT #1;AT 1,0; BRIGHT 1;T$: RETURN
299
300 REM WYSWIETLANIE
310 PRINT AT 16,0; BRIGHT 1;NR;
320 PRINT TAB 5; BRIGHT 1;D$(NR)( TO 12);
330 PRINT TAB 18; BRIGHT 1;D$(NR)(13 TO )
340 RETURN
399
400 REM KOMUNIKAT+WYBOR KLAWISZA
410 GO SUB 250
420 LET K$=INKEY$: IF K$="" THEN GO TO 420
430 LET KL=CODE (K$)-48
440 RETURN
899
900 REM KOMUNIKATY
910 LET T$="WYBIERZ !": GO SUB 400: RETURN
920 LET T$="NACISNIJ KLAWISZ !": GO SUB 400: RETURN
930 LET T$="BRAK W SLOWNIKU !": GO SUB 400: RETURN
999
1000 REM TLUMACZENIE
1010 GO SUB 220: PRINT "0-MENU"
1020 PRINT "1-ANGIELSKIE": PRINT "2-POLSKIE"
1040 GO SUB 910
1050 IF KL<0 OR KL>2 THEN GO TO 1000
1060 GO SUB 210
1070 IF KL=0 THEN RETURN
1080 IF KL=1 THEN LET N=1
1090 IF KL=2 THEN LET N=13
1100 GO SUB 230
1110 LET NR=FN I(D$(),X$,1,N)
1120 IF NR=0 THEN GO SUB 930: GO TO 1140
1130 GO SUB 300
1140 GO SUB 920: GO TO 1000
1999
2000 REM WPISYWANIE
2005 LET NR=0
2010 INPUT "NR:"; LINE N$
2020 IF N$="E" THEN RETURN
2030 IF N$="" THEN LET NR=NR+1: GO TO 2050
```

```
2040 LET NR=VAL (N$)
2050 IF NR<1 OR NR>1000 THEN  GO SUB 930: GO TO 2000
2060 GO SUB 300  INPUT "ANG:", LINE N$
2070 IF N$<>"" THEN  LET D$(NR)( TO 12)=N$
2080 GO SUB 300: INPUT "POL:"; LINE N$
2090 IF N$<>"" THEN  LET D$(NR)(13 TO )=N$
2100 GO SUB 300: GO TO 2010
2999
3000 REM ODCZYT SLOWNIKA
3010 GO SUB 240: LOAD "" DATA D$(): RETURN
3999
4000 REM ZAPIS SLOWNIKA
4010 GO SUB 240: SAVE "" DATA D$()  RETURN
```

WYDRUK 4

```
1 REM Program [illegible]
2 REM (C) Tadeusz Bagiera [illegible]
9
10 CLEAR 64769: LET ADR=[illegible]: LET SUMA=0
20 FOR I=1 TO 15
30   READ D$
40   FOR J=1 TO [illegible] STEP 2
50     LET [illegible]
60     LET [illegible]
70     LET [illegible]
80     LET [illegible]
90     LET [illegible]
100     POKE ADR,D
110     LET ADR=ADR+1
120     LET SUMA=SUMA+D
130   NEXT J
140 NEXT I
150 IF SUMA<>[illegible] THEN  PRINT "BLAD !": STOP
160 SAVE "[illegible]" CODE [illegible]
199
200 DATA "[illegible]"
210 DATA "[illegible]"
220 DATA "[illegible]"
230 DATA "[illegible]"
240 DATA "[illegible]"
250 DATA "[illegible]"
260 DATA "[illegible]"
270 DATA "[illegible]"
280 DATA "[illegible]"
290 DATA "[illegible]"
300 DATA "[illegible]"
310 DATA "[illegible]"
320 DATA "[illegible]"
330 DATA "[illegible]"
340 DATA "[illegible]"
```



[illegible]

Przy wszystkich zaletach dysków twardych typu Winchester, ich podstawową wadą pozostaje niewymienność nośnika, [redacted] to dla użytkownika liczne niemiłe konsekwencje, poczynając od kosztownego (streamer) lub czasochłonne[redacted] (dyskietki) tworzenie kopii rezerwowych, [redacted] konieczność oszczędnego gospodarowania przestrzenią na dysku, nawet wtedy, gdy dane dla każdego z zastosowań można by przechowywać [redacted] oddzielnym nośniku. Skoro jednak nie można wymieniać zawartego [redacted] hermetycznej kopercie nośnika, to czemu nie pokusić się o wymianę samej koperty z dyskiem w środku? Renomowany producent pamięci masowych Tandon zastosował w [redacted] nowym komputerze PAC [redacted] zgodnym z IBM PC/AT wymienne dyski twarde.

W przedniej ściance komputera (patrz zdjęcie) znajdują się dwie wnęki [redacted] prostokątnych, wymiennych kaset z dyskami twardymi, nazwanych Data-Pack. Każda z kaset zawiera oddzielny, hermetyzowany dysk [redacted] pojemności [redacted] MB, odporny [redacted] wstrząsy i wibracje. Wymiana kasety [redacted] jest bardziej skomplikowana od wymiany dyskietki. Ponieważ kompu[redacted] dwie stacje wymiennych dysków, kopiowanie kompletnych woluminów nie nastręcza żadnych kłopotów. Sporządzanie kompletnej kopii całego dysku [redacted] pojemności 30 MB zajmuje niewiele ponad półtorej minuty. Cena [redacted]dynczej kasety ma wynieść [redacted] RFN ok. 1000 marek, co oznacza prawie dwukrotnie mniejszy koszt zapisu jednego bitu [redacted] porównaniu z dyskami niewymiennymi. Najważniejszą zaletą zaprezentowanego rozwiązania jest jednak łatwość wymiany między poszczególnymi systemami plików danych [redacted] dużej objętości, np. kompletnych baz danych.

B. Wacławek



PAC 286 z wymiennymi kasetami dyskowymi



kąta odchylenia drążka od pozycji spoczynkowej. Tymi drugimi ze względu na charakter artykułu nie będziemy się zajmować.

Jak wiadomo, komputer składa się z jednostki centralnej, pamięci i układów peryferyjnych zwanych portami. Mikrokomputer ZX Spectrum ma wbudowany jeden port (układ ULA), który realizuje funkcje obsługi klawiatury, głośnika i koloru obrzeża ekranu (ULA realizuje także proces obsługi ekranu, ale ta część układu nie jest przez mikroprocesor traktowana jako port).

Komputer „widzi" interfejs, do którego podłącza się drążek jako port o pewnym adresie, z którego mogą być odczytywane dane. Schemat blokowy typowego interfejsu do obsługi drążka sterowego przedstawia rys. 1. Blok SEPARATORA [redacted] za zadanie oddzielić szynę danych komputera od drążka sterowego tak, aby sygnały przychodzące z drążka nie zakłócały pracy systemu. SEPARATOR przepuszcza dane (w tym wypadku w kierunku „do komputera") tylko [redacted] wypadku przyjęcia przez sygnał ZEZWOLENIE odpowiedniego stanu. Dzieje się tak, gdy mikroprocesor wystawi na szyny sterującą i adresową kombinację

STANDARDY MANIPULATORÓW DRĄŻKOWYCH DLA MIKROKOMPUTERA ZX SPECTRUM

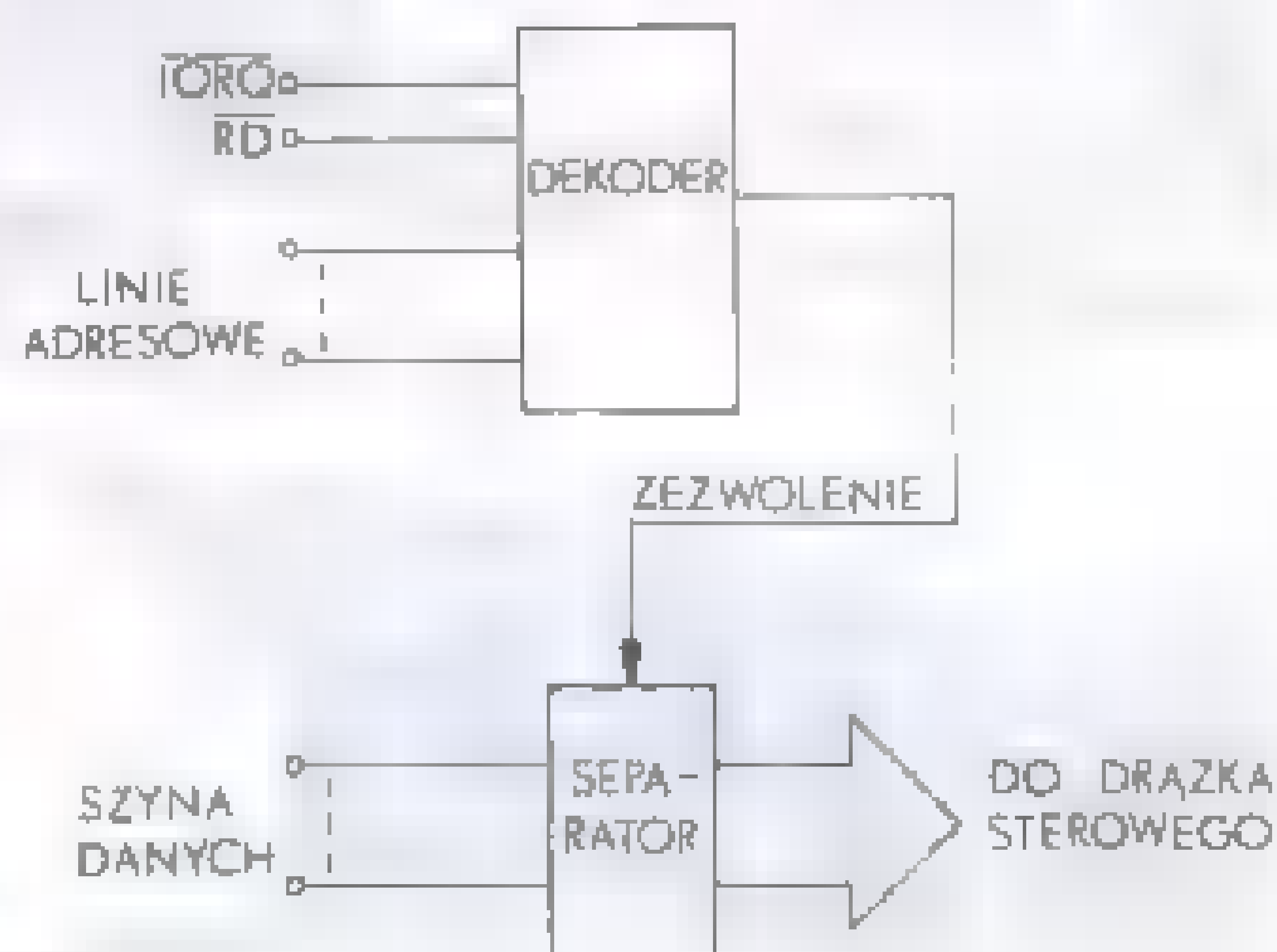
DARIUSZ ADAM PRZYGODA

Jednym z urządzeń peryferyjnych mikrokomputera ZX Spectrum jest drążek sterowy (ang. *joystick*). Mimo powszechnie panującego mniemania nie należy go uważać za element służący tylko do gier. Jego przydatność przy programach użytkowych jest [redacted] stanie ocenić każdy, kto posługiwał się takimi programami jak ART STUDIO czy ARTIST. A jeżeli właściciel używa komputera tylko do zabawy, wówczas użycie tego typu manipulatora spowoduje zmniejszenie zużycia i tak bardzo nietrwałej klawiatury.

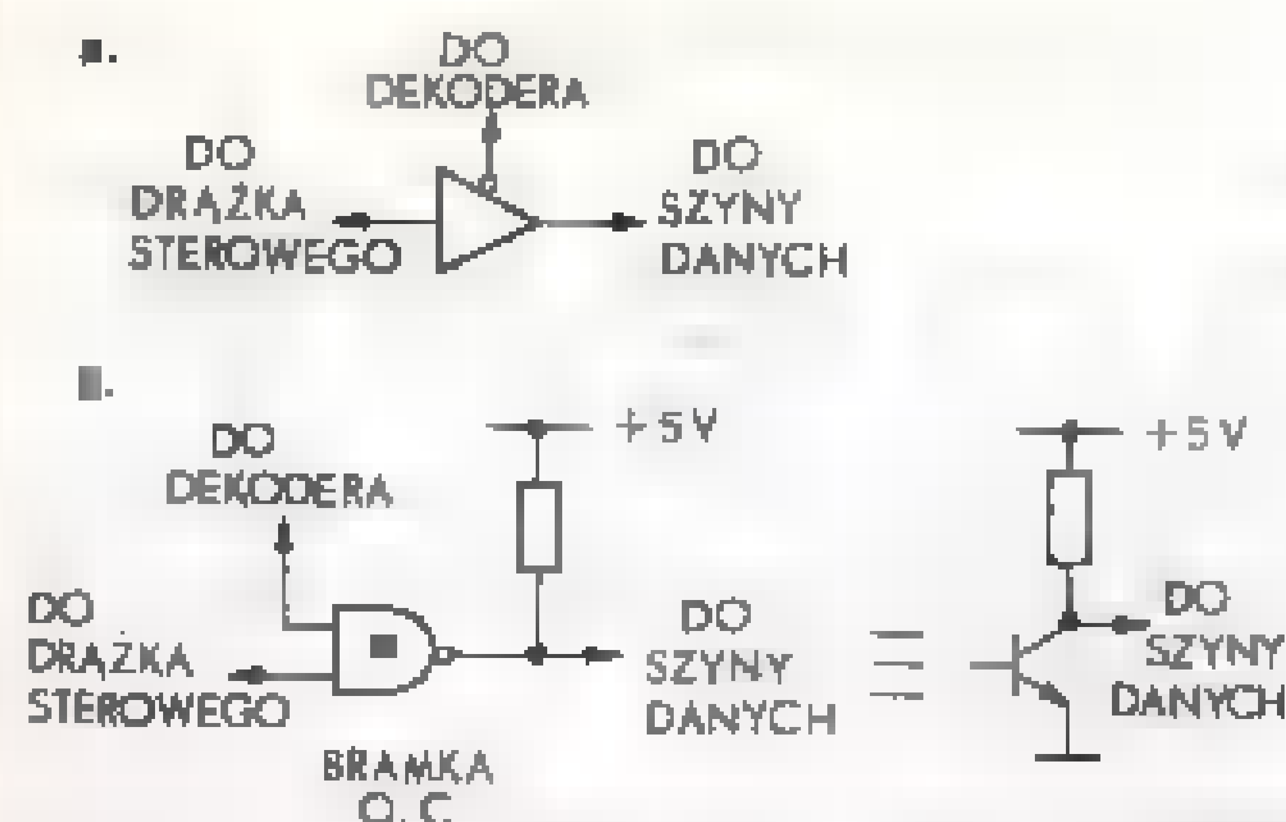
ZX Spectrum nie ma możliwości dołączenia [redacted]nipulatora bezpośrednio do komputera, ale wymaga odpowiedniego interfejsu. O ile jednak manipulatory mają w zasadzie identyczną (pod względem połączeń elektrycznych) budowę, o tyle interfejsy różnią się od siebie znacznie. Artykuł poniższy ma za zadanie dostarczyć informacji [redacted] najpopularniejszych u nas standardach i [redacted] problemach związanych z ich zastosowaniem.

Należy [redacted] zaznaczyć, że istnieją dwa typy drążków. Te, które mogą współpracować z ZX Spectrum są to drążki dwustanowe — komputer ma jedynie możliwość określenia dwóch stanów („jest — nie ma"). Istnieje poza tym druga klasa drążków, zwana analogowymi, w których istnieje możliwość interpretacji sygnałów oznaczającą wykonywanie operacji czytania danej z portu; DEKODER wykrywa tę informację i otwiera bufor sygnałem ZEZWOLENIE.

W wykonaniach praktycznych DEKODER stanowi zespół bramek tworzący odpowiedni układ kombinacyjny (czasami jest to jeden obwód scalony o większym stopniu integracji), a SEPARATOR [redacted] blok kilku (od pięciu do ośmiu) bramek z możliwością wprowadzenia wyjścia w stan wysokiej impedancji (trójstanowe lub z wyjściami typu „otwarty kolektor" — rys. 2). Spotykane są także bardziej skomplikowane konstrukcje,



Rys. 1. Schemat blokowy typowego interfejsu do obsługi drążka sterowego



Rys. 2. Rozwiązania separatorów interfejsu drążka sterowego: [illegible] z wykorzystaniem bramek trójstanowych, b) z wykorzystaniem elementów o wyjściach typu „otwarty kolektor"

jak interfejsy programowane. Chociaż są to układy bardziej skomplikowane, jednak [illegible] każdym z nich występują bloki pokazane na rys. 1.

Odczyt danych z portu wymaga od mikroprocesora Z80 (stanowiącego jednostkę centralną mikrokomputera ZX Spectrum) wykonania rozkazu IN. Mikroprocesor zawiera w swojej liście rozkazów dwa warianty tego rozkazu o formatach INA*(adres)* i IN*cel*,(C). W momencie ich wykonywania mikroprocesor podejmuje następujące akcje:

1. Na szynę adresową zostaje wystawiony *adres* (dla instrukcji INA,*(adres)* jako starszy bajt adresu podawana jest zawartość akumulatora, jako młodszy bajt adresu — wyspecyfikowany adres; dla instrukcji IN*cel*,(C) jako adres wystawiona jest zawartość pary rejestrów BC).
2. Na linie sterujące/IORQ i /RD podany zostaje stan niski.
3. Z szyny danych odczytywany jest bajt traktowany jako zawartość portu (w wypadku instrukcji INA,*(adres)* wpisywany jest on do akumulatora; w wypadku instrukcji IN*cel*,(C) umieszczany jest w miejscu specyfikowanym przez *cel)*.

Odczytany z portu bajt jest następnie analizowany. W słowie określającym stan drążka cztery bity określają wychylenie drążka w jednym z podstawowych kierunków (góra, dół, prawo, lewo), a piąty — stan przycisku (fire). Rodzaj logiki (ujemna lub dodatnia) ustalony jest w definicji standardu.

Najczęściej spotykane standardy interfejsów do współpracy z ZX Spectrum to KEMPSTON, SINCLAIR i CURSOR. Opis ich zawarty jest w tabeli 1.

Przy współpracy interfejsów z ZX Spectrum należy zwrócić uwagę na dwa często niedoceniane problemy.

Pierwszy [illegible] nich to problem jednoznacznego adresowania. Mikroprocesor Z80 ma możliwość zaadresowania 65536 różnych portów. Wymaga [illegible] użycia słowa 16-bitowego. Projektanci firmy Sinclair Research Ltd uprościli sprawę dekodowania adresu stosując tzw. dekodowanie niepełne (np. port zawarty wewnątrz układu ULA zgłasza się zawsze, jeżeli bit zerowy słowa adresowego przyjmie wartość zera logicznego, czyli przy każdym parzystym adresie portu). Różne urządzenia peryferyjne mają różnie rozwiązany problem adresacji, i trzeba uważać, aby nie kolidowały one [illegible] sobą.

Drugi z tych problemów ma o wiele bardziej subtelny charakter. Wiąże się on ze sposobem dostarczania słowa (bajtu) [illegible] szynę danych w czasie odczytu, co może być realizowane [illegible] dwa sposoby:

1. Każdy bit słowa podawany jest na szynę danych przez bramkę trójstanową uaktywnianą przez dekoder.
2. Każdy bit słowa podawany jest na szynę danych przez bramkę z wyjściem typu „otwarty kolektor"; dekoder powoduje, że poza czasem odczytu tranzystor wyjściowy bramki znajduje się [illegible] stanie odcięcia. W trakcie odczytu tranzystor ten może być nasycony (dostarczone jest zero logiczne) lub znajdować się w stanie odcięcia (dostarczona jest jedynka logiczna). W tym drugim przypadku stan logiczny „1" na szynie wymuszany jest przez opornik obciążenia tranzystora.

Powyższe zasady współpracy przedstawiono [illegible] rys. 2.

Mikrokomputer ZX Spectrum charakteryzuje się dosyć specyficzną architekturą systemową (rys. 3). Procesor wizyjny zawarty w układzie ULA do konstrukcji obrazu wymaga możliwości cyklicznego dostępu do pamięci obrazu zawartej [illegible] pamięci RAM 16 KB. Do tego samego obszaru pamięci dostęp musi mieć także mikroprocesor, który ten obszar zapełnia tworząc obraz. Obydwa urządzenia pracują w sposób w zasadzie niezależny od siebie przesyłając dane po tej samej szynie. Możliwe są następujące warianty współpracy:

1. ULA nie współpracuje z pamięcią (tworzone jest obrzeże ekranu), mikroprocesor ma wówczas wolny dostęp do wszystkich urządzeń.

TABELA 1. STANDARDY INTERFEJSÓW MANIPULATORÓW DRĄŻKOWYCH

SINCLAIR 1

Adres: 61438
Pozycja spoczynkowa: 255
Przycisk: 254

SINCLAIR 2

Adres: 63486
Pozycja spoczynkowa: 255
Przycisk: 239

KEMPSTON

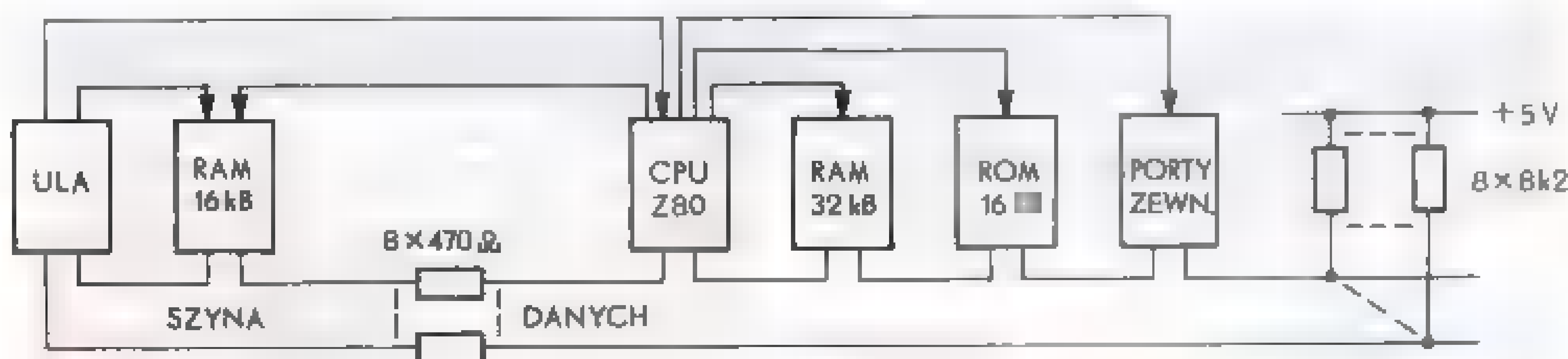
Adres: 31
Pozycja spoczynkowa: 0
Przycisk: 16

(Spotyka się adresowanie niepełne - A5 = 0)

CURSOR

Adres: 61438 i (63486)
Pozycja spoczynkowa: [illegible] i (255)
Przycisk: 254 i (255)

(Wartości odczytywane są z dwóch portów - odpowiednie dane ujęte są w nawiasy)



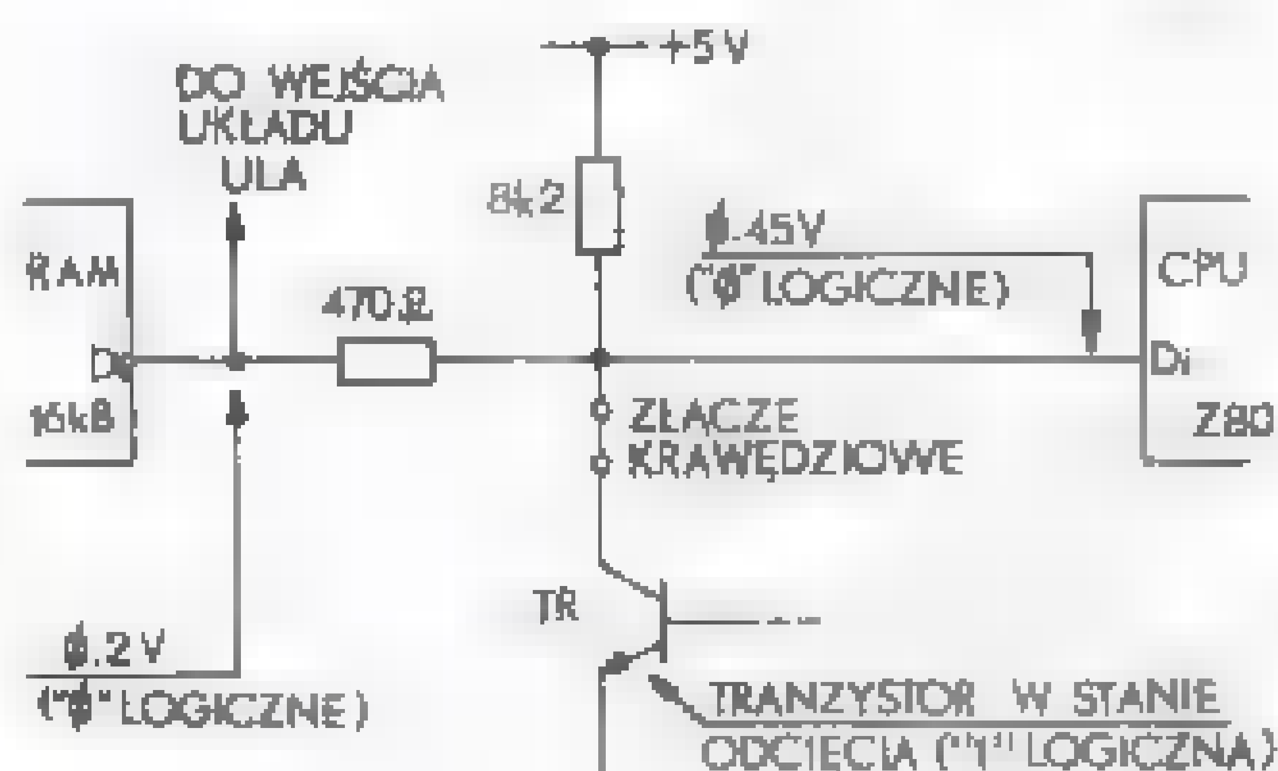
Rys. 3. Architektura systemowa mikrokomputera ZX Spectrum

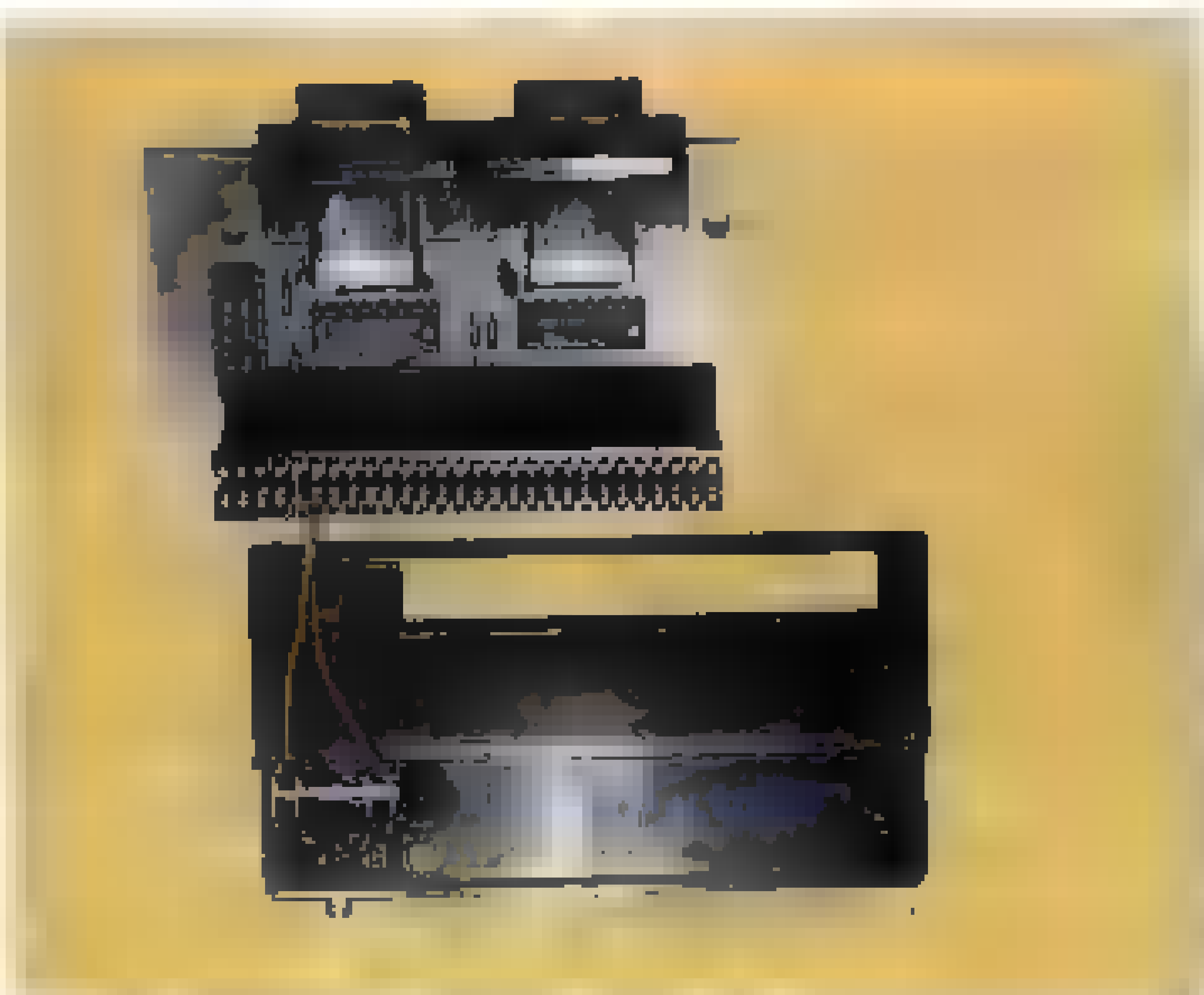
2. ULA współpracuje z pamięcią 16 KB, mikroprocesor współpracuje z dowolnym innym urządzeniem. Współpraca odbywa się współbieżnie ze względu [illegible] obecność separujących oporników 470R.
3. Zarówno ULA jak i mikroprocesor chcą współpracować z pamięcią 16 KB. Ponieważ jednoczesny dostęp do dwóch różnych komórek pamięci RAM 16 KB jest niemożliwy jedno [illegible] urządzeń musi wstrzymać pracę. Wyższy priorytet przyznano układowi ULA, który zatrzymuje przebieg zegarowy mikroprocesora na czas swojej współpracy z pamięcią 16 KB.

Rozpatrzmy teraz przebieg współpracy mikroprocesora [illegible] portem. Jeżeli będzie to rodzaj współpracy określony w punkcie 1, to bez względu na rodzaj wyjścia portu (bramki trójstanowe lub wyjścia typu „otwarty kolektor") mikroprocesor otrzyma właściwe dane. W przypadku portu z wyjściami typu „otwarty kolektor" stan wysoki na szynie zapewnią oporniki 8,2 kR.

W sytuacji opisanej w punkcie 2 prawidłowy odczyt słowa z portu będzie miał miejsce jedynie [illegible] przypadku współpracy z wyjściem trójstanowym. Przy współpracy z wyjściem typu „otwarty kolektor" stan wysoki może zostać zafałszowany przez dane pojawiające się na szynie po drugiej stronie oporników separujących. Na rys. 4 przedstawiony jest rozkład napięć na szynie danych w przypadku, gdy pamięć 16 KB dostarcza na szynę stan niski (ok. 0,2 V), interfejs — stan wysoki (tranzystor w stanie odcięcia), jednocześnie mikroprocesor dokonuje odczytu danej. Z równania dzielnika

Rys. [illegible]. Rozkład napięć przy niekorzystnym przypadku współpracy interfejsu z komputerem





oporowego wynika, [illegible] wartość napięcia [illegible] szynie danych po stronie mikroprocesora wynosić będzie tylko 0,45 V, co odpowiada wartości logicznej „0"!

Wynika z tego następujący wniosek: interfejs z wyjściami typu „otwarty kolektor" stosować można tylko wówczas, gdy mamy pewność, że ULA w czasie odczytu portu nie będzie pracować. Stan taki ma miejsce bezpośrednio po impulsie synchronizacji poziomej (ramki) obrazu — jest on tworzony przez ULA jednocześnie z impulsem przerwania podawanym [illegible] wejście INT mikroprocesora.

Jeżeli więc obsługa portu umieszczona jest w procedurze obsługi przerwania (a najczęściej tak jest), ni[illegible] [illegible] powodu do obaw. Jednakże można wskazać przykłady programów, [illegible] których obsługa interfejsu KEMPSTON zrealizowana jest [illegible] sposób niezależny od procesu obsługi przerwań (np. [illegible] pętli głównej). Jeżeli do obsługi takiego programu zastosuje się interfejs typu KEMPSTON z wyjściami typu „otwarty kolektor" (trafiają się takie na giełdach) efekt będzie daleki od zadowalającego.

Na koniec wypada przypomnieć, że ZX Spectrum jest komputerem bardzo podatnym [illegible] uszkodzenia i dlatego należy zachować dużą ostrożność przy dołączaniu interfejsów pochodzących z niepewnego źródła. Nie należy również pod żadnym pozorem dołączać lub odłączać interfejsu do komputera, gdy ten ostatni jest dołączony do sieci zasilającej. Nieprzestrzeganie tych zasad może zakończyć się poważnym uszkodzeniem komputera.

NASZ TEST:

DRUKARKA STAR NL-10

Tym razem zaprezentujemy naszym Czytelnikom test urządzenia o parametrach bardzo zbliżonych do profesjonalnych: drukarkę Star NL-10. W Polsce pracuje już bardzo wiele drukarek tego typu, a nawet w pewnym okresie możliwy był ich zakup za złotówki w sklepach górniczych. Stąd też pokusiliśmy się o dość dokładne przetestowanie tego urządzenia — poniżej przedstawimy zarówno pozytywne, jak i negatywne cechy tej drukarki, które zauważyliśmy podczas kilkumiesięcznej eksploatacji egzemplarza udostępnionego nam przez przedstawicielstwo Star w RFN i Dom Handlowy Nauki PAN.

Zaczniemy od plusów.

— **Mechanika.** Wszystkie podstawowe elementy mechaniczne drukarki zamontowane są na solidnej płycie, wykonanej z blachy odpornej na odkształcenia mechaniczne. Jest to zaleta istotna w porównaniu z drukarkami mającymi plastikowe obudowy, często gnące się już w czasie ich przenoszenia. Ta stabilność mechaniczna decyduje nie tylko o trwałości drukarki, lecz także o jakości druku, gdyż zachowane są wszystkie parametry decydujące o dokładnym prowadzeniu głowicy, co szczególnie widoczne jest przy pracy w trybie druku NLQ.

— **Głowica.** Ten podstawowy element drukarki bezpośrednio decydujący o jakości druku jest rzeczywiście mocną stroną tego modelu. Głowica charakteryzuje się dużą trwałością i precyzją wykonania. Jakość druku jest taka, że w zasadzie nawet przy 4—5-krotnym powiększeniu nie widać nierównych odległości pomiędzy śladami poszczególnych igieł, nie zauważa się także zafalowania pionowych linii w znakach (zjawisko dobrze znane użytkownikom drukarek D-100).

— **Kaseta z taśmą barwiącą.** Taśma ta charakteryzuje się znaczną trwałością: ma długość ok. 10 m i jest zwinięta w pętlę Möbiusa. Co ważniejsze, ładunek w kasecie można kilkakrotnie wymieniać bez szkody dla kasety — można też wykorzystać bez trudności taśmę z drukarki D-100. Ma ona tę samą długość i szerokość, jest także połączona w pętlę Möbiusa. Jednak jej trwałość jest dużo (około dwukrotnie) mniejsza i nie zapewnia tej jakości druku co taśma oryginalna.

— **Mechanizm prowadzenia papieru.** Zapewnia on dobrą precyzję prowadzenia papieru (ważne szczególnie przy pracy w trybie NLQ i graficznym) oraz rewers (wsteczny ruch papieru — programowy oraz sterowany z pulpitu) zarówno dla papieru z perforacją (wbudowany tzw. traktor), jak i bez niej. Nie jest to funkcja zbyt często spotykana, choć czasem bardzo przydatna.

Drukarka Star NL-10





Pięć przycisków umieszczonych w górnej części obudowy umożliwia wygodne i efektywne sterowanie trybem pracy drukarki oraz rodzajem druku bez konieczności programowego wysyłania kodów sterujących

— **Półautomatyczne wprowadzanie pojedynczych kartek papieru** — wymagane jest jedynie równe ułożenie kartki i przesunięcie dźwigni sterowania mechanizmem prowadzenia papieru w położenie 4 i z powrotem. Przydaje się to przy druku na pojedynczych arkuszach.

— **Ustawienie trybu pracy z pulpitu.** Za pomocą przycisków na obudowie drukarki możliwe jest wstępne ustawienie podstawowych rodzajów druku, jak np. 80, 96 lub 136 znaków w linii, druk wyrazisty, NLQ (również kursywa), dwa rodzaje autotestu, tryb HEX-DUMP; ustawienie lewego i prawego marginesu; dokładny przesuw papieru do przodu i do tyłu; wysuw na początek nowej strony.

— **Tryb HEX-DUMP.** Jest przydatny przy uruchamianiu różnych programów, gdy zachodzi konieczność dokładnego przeanalizowania znaków wysyłanych do drukarki. W NL-10 drukowane są nie tylko kody znaków (HEX), lecz także (o ile taka istnieje) ich reprezentacja graficzna.

— **Wewnętrzny głośniczek.** Informuje on o nieprawidłowościach pracy drukarki, potwierdza także wykonanie niektórych komend zadawanych z pulpitu.

— **Wydruk natychmiastowy.** Włączenie tego trybu pracy powoduje, że po wydrukowaniu każdego ostatniego znaku z bufora (względnie każdej nowej linii) papier jest wysuwany o kilka cm do przodu tak, że możliwy jest wygodny odczyt wydrukowanego tekstu. Przed drukiem następnych danych papier zostaje cofnięty na odpowiednią wysokość. Funkcja ta jest szczególnie przydatna podczas druku danych wysyłanych do drukarki w większych odstępach czasu.

— **Bogaty zestaw krojów pisma**, praktycznie zgodny lub szerszy od standardu drukarek Epson lub IBM. Możliwe jest łączenie wielu funkcji: np. gęstości druku i jego rodzaju (wyrazisty, kursywa, Super- i Subscript itp.).

— **Duża uniwersalność dzięki wymiennym modułom interfejsu** — dostępne są interfejsy IBM, Centronics, Commodore oraz RS-232.

— **Możliwość definiowania 94 własnych znaków**, także w trybie NLQ. Znaki wygenerowane w ten sposób mogą być drukowane we wszystkich rodzajach druku — jest to zatem najwygodniejszy sposób np. dla zaimplementowania polskich liter.

Drukarka NL-10 posiada również wiele **dodatkowych funkcji**, z których najbardziej interesujące i przydatne w pracy amatorskiej to:

1. możliwość druku znaków powiększonych 2- lub 4-krotnie,
2. makrorozkazy, ułatwiające częste sterowanie rodzajem druku,
3. rozbudowane funkcje tabulacji poziomej i pionowej,

4. formatowanie tekstu — lewo-, prawostronne lub środkowe (centrowanie tekstu).
5. znaczne możliwości graficzne, w tym także grafika 9-igłowa; kilka gęstości druku umożliwiających precyzyjne odwzorowanie różnych rodzajów grafiki na papierze.

* * *

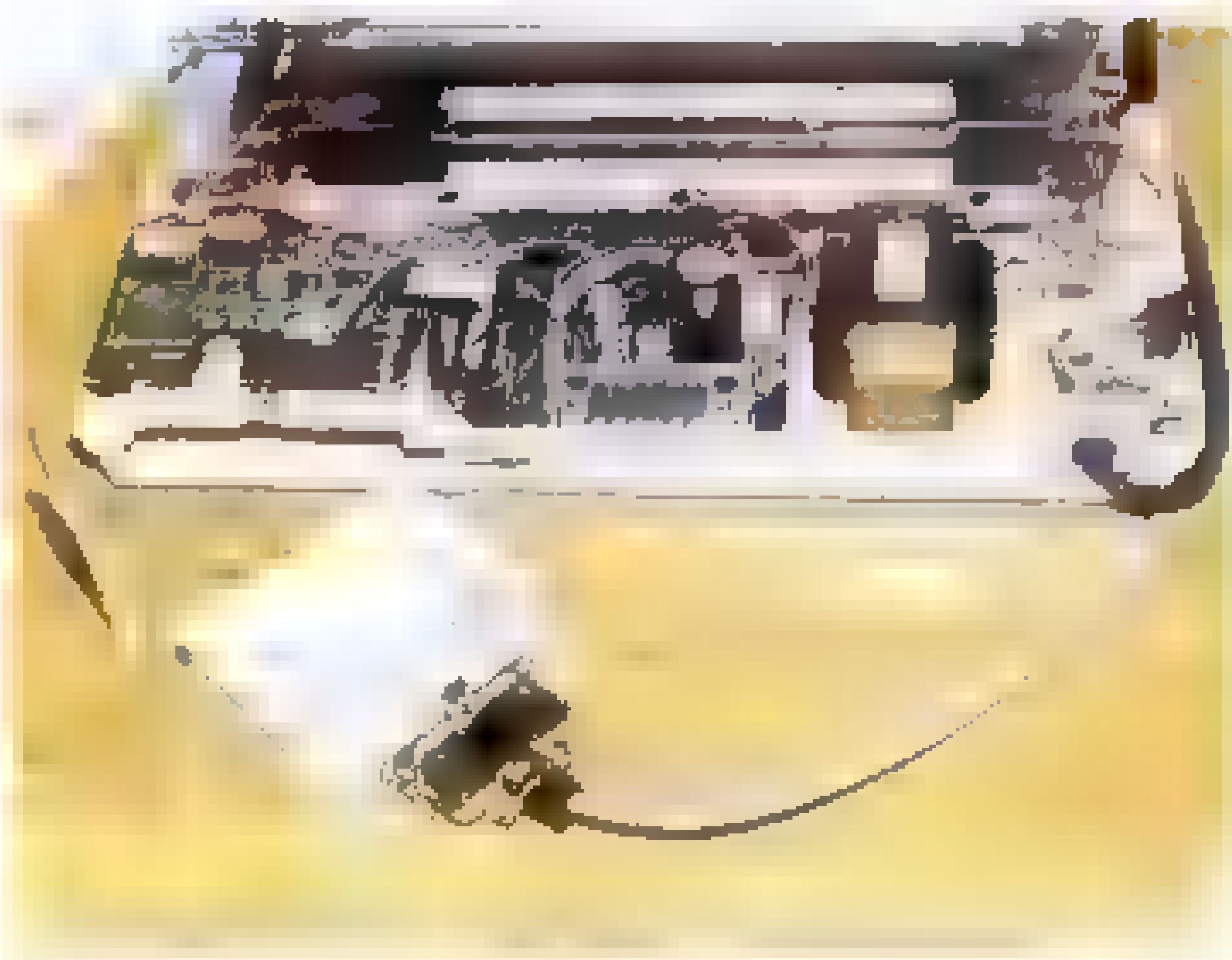
Przejdźmy teraz do omówienia zauważonych przez nas podczas eksploatacji minusów. Ich liczba nie jest zbyt wielka, lecz niektóre z tych wad są dokuczliwe i warto o nich pamiętać.

— **Blaszka na głowicy** prowadząca taśmę barwiącą ociera się o papier. Nie jest to kłopotliwe przy papierze o szorstkiej powierzchni, sprawia jednak wiele kłopotu np. przy papierze kredowym — druk jest po prostu zamazywany. Lekarstwem na tę dolegliwość jest odpowiednie wygięcie blaszki, w ten sposób, by opierała się ona o przezroczystą osłonę wykonaną z tworzywa sztucznego.

— **Dość krótki bufor danych** — przy włączonym generatorze znaków definiowanych rzędu 1 KB. Ponadto transmisja danych z komputera nie jest zbyt szybka, np. znacznie wolniejsza, niż w drukarkach Centronics.



Wymienne moduły interfejsu umożliwiają stosunkowo łatwe przystosowanie drukarki do współpracy z innym mikrokomputerem. Rozwiązanie takie jest znacznie tańsze, niż wyposażenie drukarki w kilka przełączanych interfejsów, zapewnia poza tym znacznie większą uniwersalność — można łatwo zmienić nawet generator znaków



— **Brak druku proporcjonalnego w trybie NLQ** — jest to dość poważny mankament, którego nie ma np. sprowadzona do sklepów CSH drukarka Centronics GLP II.

— **Dość głośna praca**, która jest minimalnie wygłuszona przez pokrywę mechanizmu. Szczególnie dokuczliwe przy pracy w trybie graficznym.

— **Definiowanie własnych znaków.** Ta funkcja została bardzo wyraźnie zaniedbana przez producenta. Możliwość wykorzystania tej bardzo istotnej cechy okupiona jest kilkoma bardzo ważnymi mankamentami:

1. Znaków definiowanych jest tylko 94 — nie można zdefiniować nie tylko znaku o kodzie 127 (DEL — występuje to też w innych drukarkach), lecz również znaku o kodzie 32 (kod spacji) — to już w innych drukarkach spotykane jest raczej rzadko. W instrukcji obsługi mankament ten opisany jest dopiero przy szczegółowym opisie wykorzystania znaków definiowanych — w danych technicznych umieszczonych na wstępie informacji tej nie ma.

2. Każdorazowa zmiana generatora z normalnego na definiowalny i odwrotnie powoduje wydruk bufora druku(!). W przypadku prostego druku bez funkcji specjalnych powoduje to tylko bardzo znaczne spowolnienie druku (od kilku do kilkunastu razy w zależności od ilości zmian generatora w linii). W celu minimalizacji tego efektu konieczne staje się stosowanie specjalnych „kruczków". Ogranicza to w pewien sposób możliwość zaprogramowania polskich liter bez uszczuplenia standardowego generatora znaków. Dodajmy, że m.in. drukarka Centronics GLP II wady tej nie posiada.

Wada ta powoduje jeszcze inne, bardzo przykre następstwa: przesuwanie wydruku w pionie przy druku znaków powiększonych (oczywiście przy zmianie generatora). Konieczne wówczas byłoby zastosowanie programowego cofania papieru. Nie działa także w takim przypadku funkcja automatycznego formatowania — powstają „dziury" wewnątrz tekstu, co rzecz jasna uniemożliwia wykorzystanie go.

3. W starszych wersji interfejsu Centronics (wersje 1.4 i wcześniejsze) następowała częściowa utrata danych przy operowaniu klawiszami przesuwu papieru przy zapełnionym buforze danych i wykryciu końca papieru (oczekiwanie na nową kartkę). Powodowało to utratę części tekstu, a także przekłamania rejestru położenia głowicy, co doprowadzało do jej „dobijania" do ograniczników i w konsekwencji zatrzymanie drukarki na błędzie. Na szczęście w nowszych wersjach (1.5 i 1.6) wadę tę wyeliminowano. Występowała ona jednak m.in. w drukarkach sprowadzonych do sklepów górniczych (dodatkowo były one przystosowane do napięcia 240 V, w niektórych sytuacjach może to u nas spowodować zakłócenia pracy).

* * *

Reasumując należy stwierdzić, że mimo kilku dość istotnych wad drukarka Star NL-10 jest urządzeniem dobrym, nadającym się do wykorzystania przez półprofesjonalnych i profesjonalnych informatyków. NL-10 polecić można wszystkim tym, którzy myślą o poważnym wykorzystaniu komputera. Jest to drukarka tym bardziej atrakcyjna, że jej cena w RFN wynosi ok. 550 DM, a zatem kosztuje ona niewiele więcej, niż proste drukarki amatorskie o znacznie gorszych parametrach.

Grzegorz Zało



Kangur z Holandii

Supermikrokomputerami nazywamy jednostki pośrednie pomiędzy mikro- i mini-komputerami. Do tej rodziny należy Kangaroo Model 350, wyposażony w mikroprocesor Clipper. Wewnątrz komputera mieści się 4 MB pamięci RAM, którą można rozszerzyć do 16 MB. Jednostka centralna posiada stację dysków 5,25 cala o pojemności 1,2 MB oraz dysk twardy 70 MB. Archiwizowanie dużych ilości informacji zapewnia szybka pamięć taśmowa (ang. tape streamer) na kasetach mieszczących 60 MB.



Komputer pracuje pod kontrolą systemu CLIX, wzorowanego na UNIX V. Obsługuje jednocześnie do 24 użytkowników, osiągając średnią wydajność 4.5—5 MIPS. Został skonstruowany w Eindhoven — mieście będącym dla krajów Beneluxu tym, czym Krzemowa Dolina dla USA.

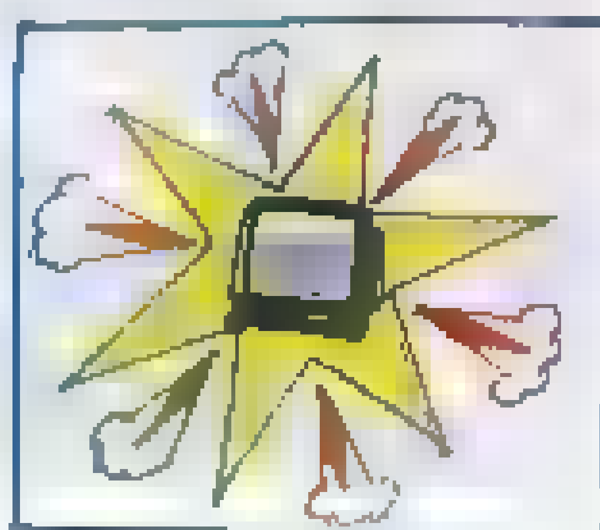
Kolorowy Hercules

Firma Hercules Computer Technology wyprodukowała kolejny model karty graficznej — tym razem kolorowej. Stanowi ona rozwinięcie koncepcji znanych z modelu Hercules Plus (o którym pisaliśmy w „InforMiku" 3 1987) i na przykład umożliwia samodzielne zdefiniowanie 3072 znaków. Rozdzielczość ekranu wynosi 720×348 punktów; można jednocześnie użyć szesnastu kolorów z palety 64. Do każdego egzemplarza jest dodawana dyskietka firmowa z handlerami umożliwiającymi wykorzystanie walorów nowej karty przez programy Auto CAD, Lotus 1-2-3, Syphony i MS Windows.



Gwiezdny wojownik

Sky Warrior jest procesorem wektorowym (wykonującym operacje na wielu elementach tablicy równocześnie) zwiększającym wydajność mini-komputerów VAX. W ciągu sekundy wykonuje 15 milionów zmiennoprzecinkowych operacji arytmetycznych na liczbach 32- lub 64-bitowych. Pojemność pamięci podręcznej wynosi, zależnie od wersji, 64 lub 256 kilobajtów. Procesor może zaadresować 10 megabajtów pamięci operacyjnej.



Efektywne wykorzystanie procesorów wektorowych jest bardzo trudne, wymaga uwzględnienia kilku procesów obliczeniowych przeprowadzanych równocześnie. Sky Computers Inc., producent procesora, opracował dla niego pakiet procedur realizujących obliczenia arytmetyczne.

Wojna o dyski

Czołowe światowe firmy produkują dyski twarde 3,5 cala o coraz większej pojemności. MK-130F firmy Toshiba mieści 53 MB danych. Czas dostępu wynosi 25 ms, szybkość wymiany informacji z komputerem równa jest pięciu megabodom. Średni czas pracy bezawaryjnej (MTBF ang. Mean Time Between Failures) wynosi [illegible] 000 godzin.



Seagate oferuje całą rodzinę dysków, które mogą być zapisane w formacie MFM lub RLL. Ich pojemność wynosi od 20 do 45 MB, czas dostępu 30 ms.

Firma Panasonic skonstruowała dysk JU 127, którego pojemność wynosi 53 MB. Czas dostępu do informacji sięga 40 ms, jednak na zmianę odczytywanej ścieżki potrzeba tylko 9 ms. Dane są przesyłane z szybkością 5 Mbit/s (megabodów). Jak podaje producent, gęstość zapisu informacji wynosi 13171 bpi (bitów na cal) — zatem pojedynczy bit zajmuje na ścieżce odcinek 1.9 mikrometra. Średni czas pracy bezawaryjnej dysku wynosi 20 000 h.

Ekran plazmowy

W numerze 2/87 pisaliśmy o monitorach z płaskim ekranem. Oprócz opisanego wyświetlacza LCD istnieją również ekrany plazmowe, często stosowane w komputerach przenośnych. Do tej rodziny należy ekran firmy Fujitsu o rozdzielczości 640×400 punktów, z których każdy ma średnicę 0,33 mm. Obraz o rozmiarach 211×132 mm odpowiada przekątnej 9,8 cala. Cały ekran o wymiarach 300×200×27 mm waży 1,5 kg.

Wyświetlane napisy i rysunki mają pomarańczową barwę. Pole widzenia przekracza 120 stopni — znacznie więcej, niż w wyświetlaczach LCD. Jasność ekranu wynosi 150 cd/m², zaś kontrast 20 : 1.

Postęp w konstruowaniu pamięci półprzewodnikowych

Od dłuższego czasu w dziedzinie pamięci dynamicznych o dużych pojemnościach przodują firmy japońskie. W lutym 1987 r. wytwórnia NTT przedstawiła prototyp pamięci RAM o pojemności 16 megabitów. Dzięki ograniczeniu rozmiarów elementów do 0,5—0,8 mikrometra na powierzchni 1 cm zmieszczono około 35 milionów tranzystorów.

Nie wiadomo, kiedy rozpocznie się produkcja seryjna, natomiast układy 4 MB firmy NMB są wytwarzane już od wiosny 1988 roku. Zastosowanie technologii CMOS 0,8 mikrometra ma zapewnić czas dostępu rzędu 100 ns.

W czerwcu 1987 r. do klubu wytwórców pamięci 1-megabitowych dołączyła firma Sony. Obecnie pamięci są wykonywane w technologii NMOS 1,2 mikrometra, jednakże Sony przymierza się już do technologii CMOS. Czas dostępu do danych wynosi zaledwie 70 ns, co umożliwia zastosowanie elementów w układach telewizji cyfrowej.

Elektroniczne sztalugi

Francuska firma Cetia skonstruowała terminal do prac graficznych (ang. graphics workstation) 4—25, którego sercem jest mikroprocesor 68020 z koprocesorem 68881. Pamięć terminala stanowi 512 KB RAM, rozszerzalne do 2 megabajtów. W zależności od pojemności pamięci można użyć



16 lub 256 kolorów z palety 16,67 miliona. Rozdzielczość ekranu wynosi 1280×1024 punkty.

Moc obliczeniowa 68020, nawet z koprocesorem arytmetycznym, nie wystarczy do animacji komputerowej czy poważnych prac inżynierskich. Większość obliczeń wykonuje więc mini-komputer VAX lub MicroVAX, z którym terminal jest połączony poprzez interfejs RS-232C.

Cetia 4-25 może w jednym z trybów pracy naśladować popularny na Zachodzie terminal Tektronix 41xx.

Kolejny standard!

Hitachi produkuje dyski o średnicy 8,8 cala oznaczane jako DK 815-10. Pojemność dysku niesformatowanego wynosi 1067 MB, natomiast sformatowanego 852 MB — dwukrotnie więcej, niż dla dysku 8-calowego tej samej firmy. Czas dostępu wynosi 15 milisekund, informacje są przesyłane przez interfejs ESMD z szybkością 2460 Kilobodów.

Dyski o niemal identycznych parametrach wytwarza również NEC.

CD-ROM

PHILIPS jest jednym z pierwszych producentów odtwarzaczy płyt kompaktowych, nic dziwnego, iż działa również na rynku pamięci optycznych. Stacja CM 201 służy do odczytywania informacji z płyt o średnicy 5,25 cala. Pojemność płyty wynosi 600 megabajtów. Czas dostępu — 0.4 sekundy — jest znacznie większy, niż dla dysków magnetycznych, ale pamięci optyczne innych producentów mają czas dostępu podobny. Połączenie z komputerem stanowi szybki interfejs szeregowy.

Kontroler stacji został wyposażony w układy automatycznej korekcji błędów — na 10 do 10 odczytanych bitów tylko jeden bit w informacji przesłanej do komputera może być błędny.

Pamięć zamiast wideo

Amerykańska firma Honeywell wyprodukowała system pamięci taśmowej VLDS, który przechowuje 6,2 gigabajta informacji w typowej kasecie wideo T-120 systemu VHS. Zastosowano kod zapisu Reed-Solomon zapewniający częstość występowania błędów niższą od 10. Dane są zapisywane w blokach o długości 32 KB z nagłówkami przy gęstości dwóch tysięcy bitów na milimetr. Szybkość przesyłu informacji wynosi 6 MB/s, przewinięcie i przeszukanie całej kasety trwa zaledwie 90 sekund. Tak duża prędkość przesuwu taśmy bardzo ją niszczy — VLDS nie jest przeznaczony do bieżącej obsługi systemu komputerowego, lecz do archiwizowania dużej ilości danych.

Opracował: **Adam Nowicki**

S E M I N A R I U M
I N F O R M I K A

ASEMBLER GENS3

Część [illegible]

Definiowanie makroinstrukcji

Jak już wspomnieliśmy w poprzednim odcinku, wersja asemblera GENS3M2 została wyposażona dodatkowo w możliwość definiowania makroinstrukcji. Makroinstrukcje możemy traktować funkcjonalnie jako rozszerzenie dostępnego dla programisty zbioru instrukcji programowych. Zasadnicza różnica między nimi polega na tym, że zbiór instrukcji programowych jest stały i ich nazwy są zastrzeżone w wewnętrznym słowniku nazw, natomiast instrukcje makro są definiowane przez programistę. Nazwy makroinstrukcji i odpowiadające im ciągi instrukcji programowych są umieszczane w wewnętrznym obszarze pamięci nazywanym w przypadku GENS3M2 „Macro buffer". Ponadto, o ile instrukcje programowe są tłumaczone na pojedyncze rozkazy maszynowe, to instrukcjom makro mogą odpowiadać dłuższe ciągi rozkazów maszynowych. Dodatkowo można przekazywać parametry do wnętrza makroinstrukcji. Samo tłumaczenie makroinstrukcji w trakcie asemblacji może być warunkowe, ponieważ dopuszcza się stosowanie pseudoinstrukcji asemblacji warunkowej wewnątrz definicji makroinstrukcji.

Definicja makroinstrukcji składa się z następujących elementów:

1. Nazwy makroinstrukcji umieszczonej w polu etykiety.
2. Dyrektywy MAC umieszczonej w polu operacji.
3. Wykazu parametrów formalnych umieszczonego w polu operacji.
4. Ciągu instrukcji programowych.
5. Dyrektywy ENDM umieszczonej w polu operacji.

Uwagi:

1. Nazwa makroinstrukcji jest obowiązkowa. Identyfikowane są maksymalnie 4 znaki nazwy. Brak nazwy spowoduje wstrzymanie asemblacji i wystąpienie komunikatu ERROR [illegible].
2. Dyrektywa MAC jest obowiązkowa.
3. Wykaz parametrów formalnych jest opcjonalny. Jeśli w definicji makroinstrukcji nie wykorzystujemy parametrów jest to oczywiste, jeśli jednak wykorzystujemy parametry to zaleca się ze względu na dokumentację programu opisywanie parametrów odpowiednimi do ich znaczenia tekstami. Celowo zostało tu użyte słowo „tekstami", gdyż nie zostaną one umieszczone tak jak nazwy w tablicy symboli.

Maksymalnie lista parametrów może zawierać 32 elementy. Poszczególne parametry oddzielamy znakiem przecinka. Wewnątrz definicji makroinstrukcji parametry są identyfikowane przez użycie znaku równości z numerem kolejnego parametru od 1 do 32. Na tak zidentyfikowanym parametrze jest dopuszczalne wykonywanie dostępnych w GENS3 operacji arytmetycznych.

4. W ciągu instrukcji programowych zawartych wewnątrz definicji makroinstrukcji nie może wystąpić definicja makroinstrukcji (Komunikat ERROR [illegible]). Nie może również wystąpić wywołanie makroinstrukcji (Komunikat ERROR [illegible]).





Wywoływanie makroinstrukcji

Zdefiniowane makroinstrukcje wywołujemy w programie przez podanie w polu operacji nazwy identyfikującej makroinstrukcję i podanie ewentualnego wykazu parametrów aktualnych w polu argumentów. Parametry wywołania mogą być w ogólnym przypadku wyrażeniami arytmetycznymi. W pierwszym przebiegu asemblacji asembler umieszcza definicje makroinstrukcji w buforze makroinstrukcji. O ile pierwszy przebieg jest prawidłowy, w drugim przebiegu w momencie napotkania wywołania makroinstrukcji analizuje linie programowe umieszczone w definicji makroinstrukcji i zastępuje odpowiednie parametry formalne parametrami aktualnymi. Tym samym pojedyncza makroinstrukcja programu źródłowego jest rozwijana w ciąg linii programowych.

Należy tu zaznaczyć, że w podstawowej postaci wydruku asemblacji rozwinięcie to nie występuje. Listowana jest jedynie linia programowa zawierająca wywołanie makroinstrukcji. Jeżeli chcielibyśmy zobaczyć jak rozwijana jest makroinstrukcja należy użyć dyrektywy M + oraz dyrektywy M — dla ewentualnego wyłączenia listowania rozwinięcia. Taka sytuacja występuje w przytoczonym poniżej wydruku przykładowego programu źródłowego, gdzie ze względów dydaktycznych uwidoczniono rozwinięcia. Oczywiście w praktyce nie musimy tego stosować, gdyż możliwość definiowania makroinstrukcji jest jednym ze środków upraszczających programowanie, a o wiele trudniej pracuje się nad programem zawierającym dużą ilość nieistotnych dla problemu informacji. Należy tu podkreślić, że stosowanie makroinstrukcji może w znacznym stopniu zmniejszyć objętość programu źródłowego, natomiast objętość programu wynikowego może w ostatecznym rozrachunku wzrosnąć. Jest to na ogół powodowane następującymi przyczynami:

1. Stosowaniem makroinstrukcji o zbyt wielu liniach programowych, co jest charakterystyczne dla początkujących programistów.

2. Próbami zastępowania całych procedur o znacznych rozmiarach makroinstrukcjami, co jest charakterystyczne dla sytuacji, w której programista próbuje „przerobić" asembler [redacted] język programowania wyższego rzędu.

Zilustrujemy to na przykładzie kilku definicji makroinstrukcji zdefiniowanych w poniższym przykładzie.

```
10 ;ASEMBLER GENS3M2
20 ;PRZYKLADY DEFINICJI MAKRO
30 ;ORAZ ICH WYKORZYSTANIE
40 ;---------------------------
50             ORG   60000
60 *M+
70             JP    START
80 ZMIEN1      DEFS  2
90 ZMIEN2      DEFS  2
100 ZMIEN3     DEFW  32768
110 ZMIEN4     DEFS  2
120 ZMIEN5     DEFS  2
130 ZMIEN6     DEFS  2
140 BUFOR      DEFS  16
150 STALA1     EQU   16384
160 STALA2     EQU   50000
170 ;---------------------------
180 ;DEFINICJE
190 ;---------------------------
200 #LDH       MAC
210            LD    E,(HL)
220            INC   HL
230            LD    D,(HL)
240            INC   HL
250            ENDM
260 ;---------------------------
270 #LIX       MAC   ZMIEN,WSKAZN
280            LD    L,(IX+=1)
290            LD    H,(IX+1+=1)
300            LD    (=0),HL
310            ENDM
[redacted] ;---------------------------
330 LET2       MAC   ZMIEN,STALA
340            LD    HL,=1
350            LD    (=0),HL
360            ENDM
370 ;---------------------------
380 #SU2       MAC   ZMIEN,ZMIEN
390            LD    HL,(=0)
400            LD    DE,(=1)
410            XOR   A
420            SBC   HL,DE
430            LD    (=0),HL
440            ENDM
450 ;---------------------------
460 MOVE       MAC   OD,DO,ILE,JAK
470            LD    HL,=0
480            LD    DE,=1
490            LD    BC,=2
500            IF    =3
510            LDIR
520            ELSE
530            LDDR
540            END
550            ENDM
560 ;---------------------------
570 START
580            LD    HL,ZMIEN3
590            #LDH
600            LD    (ZMIEN4),DE
610
620            LD    IX,BUFOR
630            #LIX  ZMIEN5,0
640            #LIX  ZMIEN6,2
650
660            LET2  ZMIEN1,100
670            LET2  ZMIEN2,STALA1
680            #SU2  ZMIEN3,ZMIEN2
690            MOVE  0,#4000,2048,1
700            MOVE  #4000+2048,STALA2,2048,0
710            RET
720 ---------------------------
```

```
*HISOFT GENS3M2 ASSEMBLER*
        ZX SPECTRUM

Copyright (C) HISOFT 1983,4
All rights reserved

Pass 1 errors: 00

                    10 ;ASEMBLER GENS3M2
                    20 ;PRZYKLADY DEFINICJI MAKRO
                    30 ;ORAZ ICH WYKORZYSTANIE
                    40 ;---------------------------
EA60                50            ORG   60000
                    60 *M+
EA60 C37FEA         70            JP    START
EA63                80 ZMIEN1     DEFS  2
EA65                90 ZMIEN2     DEFS  2
EA67 0080          100 ZMIEN3     DEFW  32768
EA69               110 ZMIEN4     DEFS  2
EA6B               120 ZMIEN5     DEFS  2
EA6D               130 ZMIEN6     DEFS  2
EA6F               140 BUFOR      DEFS  16
4000               150 STALA1     EQU   16384
C350               160 STALA2     EQU   50000
                   170 ;---------------------------
                   180 ;DEFINICJE
                   190 ;---------------------------
EA7F               200 #LDH       MAC
EA7F               210            LD    E,(HL)
EA7F               220            INC   HL
EA7F               230            LD    D,(HL)
EA7F               240            INC   HL
EA7F               250            ENDM
                   260 ;---------------------------
EA7F               270 #LIX       MAC   ZMIEN,WSKAZN
EA7F               280            LD    L,(IX+=1)
EA7F            [redacted]        LD    H,(IX+1+=1)
EA7F               300            LD    (=0),HL
EA7F               310            ENDM
                [redacted] ;---------------------------
EA7F               330 LET2       MAC   ZMIEN,STALA
EA7F               340            LD    HL,=1
EA7F               350            LD    (=0),HL
EA7F               360            ENDM
                   370 ;---------------------------
EA7F               380 #SU2       MAC   ZMIEN,ZMIEN
EA7F               390            LD    HL,(=0)
EA7F               400            LD    DE,(=1)
EA7F               410            XOR   A
EA7F               420            SBC   HL,DE
EA7F               430            LD    (=0),HL
EA7F               440            ENDM
                   450 ;---------------------------
EA7F               460 MOVE       MAC   OD,DO,ILE,JAK
EA7F               470            LD    HL,=0
EA7F               480            LD    DE,=1
EA7F               490            LD    BC,=2
EA7F               500            IF    =3
EA7F               510            LDIR
EA7F               520            ELSE
EA7F               530            LDDR
EA7F               540            END
EA7F               550            ENDM
                   560 ;---------------------------
                   570 START
EA7F 2167EA        580            LD    HL,ZMIEN3
EA82               590            #LDH
EA82 5E                           LD    E,(HL)
EA83 23                           INC   HL
EA84 56                           LD    D,(HL)
EA85 23                           INC   HL
EA86 ED5369EA      600            LD    (ZMIEN4),DE
                   610
EA8A DD216FEA      620            LD    IX,BUFOR
EA8E               630            #LIX  ZMIEN5,0
EA8E DD6E00                       LD    L,(IX+=1)
EA91 DD6601                       LD    H,(IX+1+=1)
EA94 226BEA                       LD    (=0),HL
EA97               640            #LIX  ZMIEN6,2
EA97 DD6E02                       LD    L,(IX+=1)
EA9A DD6603                       LD    H,(IX+1+=1)
EA9D 226DEA                       LD    (=0),HL
                   650
EAA0               660            LET2  ZMIEN1,100
EAA0 216400                       LD    HL,=1
EAA3 2263EA                       LD    (=0),HL
EAA6               670            LET2  ZMIEN2,STALA1
EAA6 210040                       LD    HL,=1
EAA9 2265EA                       LD    (=0),HL
EAAC               680            #SU2  ZMIEN3,ZMIEN2
EAAC 2A67EA                       LD    HL,(=0)
EAAF ED5B65EA                     LD    DE,(=1)
EAB3 AF                           XOR   A
EAB4 ED52                         SBC   HL,DE
EAB6 2267EA                       LD    (=0),HL
EAB9               690            MOVE  0,#4000,2048,1
EAB9 210000                       LD    HL,=0
EABC 110040                       LD    DE,=1
EABF 010008                       LD    BC,=2
0001                              IF    =3
EAC2 EDB0                         LDIR
EAC4                              ELSE
EAC4                              LDDR
EAC4                              END
EAC4               700            MOVE  #4000+2048,STALA2,2048,0
EAC4 210048                       LD    HL,=0
EAC7 1150C3                       LD    DE,=1
EACA 010008                    [redacted]    BC,=2
0000                              IF    =3
EACD                              LDIR
EACD                              ELSE
EACD EDB8                         LDDR
EACF                              END
EACF C9            710            RET
                   720 ---------------------------
```

```
Pass 2 errors: 00

$LDH    EA7F     $LDX    EA7F
$SUB    EA7F     BUFOR   EA6F
LET2    EA7F     MOVE    EA7F
STALA1  4000     STALA2  C350
START   EA7F     ZMIEN1  EA63
ZMIEN2  EA65     ZMIEN3  EA67
ZMIEN4  EA69     ZMIEN5  EA6B
ZMIEN6  EA6D

Table used:    196 from  2000
```

Przykłady zastosowania makroinstrukcji

Przykład 1

Często stosowaną sekwencją instrukcji mającą na celu załadowanie pary rejestrów DE zawartością kolejnych dwóch komórek pamięci adresowanych przez parę rejestrów HL zastępujemy bezparametrową makroinstrukcją $LDH. Nazwę dla łatwiejszego zapamiętania skojarzono z nazwą instrukcji asemblera, poprzedzając ją dla wyróżnienia znakiem $.

Przykład 2

Zadaniem drugiej makroinstrukcji jest nadanie dwubajtowej zmiennej wartości znajdujących się pod dwoma kolejnymi adresami pamięci adresowanymi za pomocą rejestru indeksowego IX. W wywołaniu tej makroinstrukcji podajemy w pierwszym parametrze nazwę zmiennej, a w drugim wskaźnik. Jeśli przyjąć że do rejestru IX wpisaliśmy na stałe adres pierwszej komórki obszaru wypełnionego 128 liczbami to makroinstrukcja ta umożliwia w istocie pobieranie danych z 64-elementowej tablicy liczb dwubajtowych.

Przykład 3

Makroinstrukcja nadaje zmiennej dwubajtowej będącej pierwszym parametrem wartość będącą drugim parametrem. Jest to instrukcja w zasadzie identyczna ze znaną instrukcją podstawienia, stąd dla przykładu wybrano taką nazwę. Nasuwa się tu ogólniejsza uwaga. Na ogół staramy się używać polskiego nazewnictwa etykiet czy procedur w programach źródłowych. Wydaje się jednak, że w sytuacji ograniczenia się do czteroliterowych nazw można, ze względu na bezpośrednie skojarzenia z językami programowania, odstąpić czasem od tej reguły.

Przykład 4

Wykonanie odejmowania dwóch liczb dwubajtowych wymaga wykonania typowej sekwencji instrukcji. Widać, że próba zastąpienia jej procedurą nie daje żadnych oszczędności w programie źródłowym, ani w wynikowym. Powodem jest konieczność przekazania parametrów. Użycie makroinstrukcji upraszcza istotnie postać programu źródłowego, ponadto nie powoduje wzrostu objętości programu wynikowego.

Przykład 5

Motywacja, jak w poprzednim przykładzie. Ponadto makroinstrukcja posiada dwa rozwinięcia. Pierwsze trzy linie są tłumaczone bezwarunkowo, a następnie w zależności od czwartego parametru wywołania uwzględniona zostanie instrukcja LDIR, jeśli parametr był różny od zera lub instrukcja LDDR, jeśli parametr był równy zero.

Przedstawione powyżej przykłady nie są zbyt wyszukane i nie wyczerpują możliwości stosowania makroinstrukcji. Przytoczenie ich miało na celu przybliżenie programiście dodatkowych możliwości makroasemblera. Praktyczne wykorzystanie makroinstrukcji będzie ostatecznie zależało od stopnia zaawansowania programisty i złożoności programu.

Dodatkowe zlecenia edytora i asemblera

Po uruchomieniu asemblera GENS3M2 [redacted] trzeba podawać rozmiarów bufora dla asemblacji z taśmy programów nagranych za pomocą dyrektywy T. W tej wersji asemblera rozmiar bufora jest automatycznie ustalany na 256 bajtów. Jeśli zamierzamy zmienić rozmiar bufora na większy (256 to minimum!) należy użyć zlecenia C. Po pytaniu „Include buffer?" należy podać żądany rozmiar bufora w bajtach. Naciśnięcie klawisza ENTER spowoduje przyjęcie wartości 256. Nastąpi teraz pytanie „Macro buffer?", po którym podajemy w bajtach rozmiar bufora dla makroinstrukcji. Naciśnięcie klawisza ENTER spowoduje przyjęcie wartości równej zero. Jeżeli po wykonaniu zlecenia asemblacji wyprowadzony zostanie komunikat „No Macro Space", należy nagrać program źródłowy, powiększyć rozmiar bufora i ponownie wczytać program źródłowy. Bezpośrednie powiększenie rozmiarów bufora powoduje utratę programu źródłowego, ponieważ oba obszary są rezerwowane w pamięci operacyjnej przed programem źródłowym. Dla orientacji podajemy, że w przytoczonym przykładzie wszystkie definicje makroinstrukcji zajmują razem 220 bajtów. Ponieważ w wersji GENS3 zleceniem C dokonywaliśmy tłumaczenia programów wygenerowanych przez MONS3, w wersji asemblera GENS3M2 została przeznaczona do tego celu zamiast litery C litera O.

Wersja asemblera GENS3M2 została wyposażona dodatkowo w zlecenia umożliwiające współpracę z pamięcią masową w postaci mikrodrajwów. Format tych zleceń jest następujący:

G , , 1: nazwa

Zleceniem tym wczytujemy program źródłowy zapisany na kasecie mikrodrajwu. Jeżeli w pamięci operacyjnej znajduje się już tekst programu źródłowego aktualnie wczytany program zostanie umieszczony za istniejącym, a całość otrzyma numerację linii od jeden [redacted] skokiem jeden.

Pm , n , 1: nazwa

Zleceniem tym zapisujemy program źródłowy na kasecie mikrodrajwu: m oznacza numer pierwszej linii, [redacted] oznacza numer ostatniej linii.

H , , 1: nazwa

Zleceniem tym weryfikujemy program źródłowy zapisany [redacted] kasecie mikrodrajwu z programem istniejącym w pamięci operacyjnej. W trakcie weryfikacji na ekranie listowane są numery kolejno poprawnie zweryfikowanych rekordów, ponieważ program źródłowy jest zapisany w postaci zbioru sekwencyjnego.

W przytoczonych zleceniach liczba jeden oznacza numer stacji (może być od 1 do 8). Nazwa może być maksymalnie 10-elementowym łańcuchem znaków ASCII.

Możliwość współpracy GENS3M2 zostanie omówiona w następnym odcinku. Wymagać będzie ona podobnego jak w przypadku GENS3 programu obsługowego.

Tadeusz [redacted]



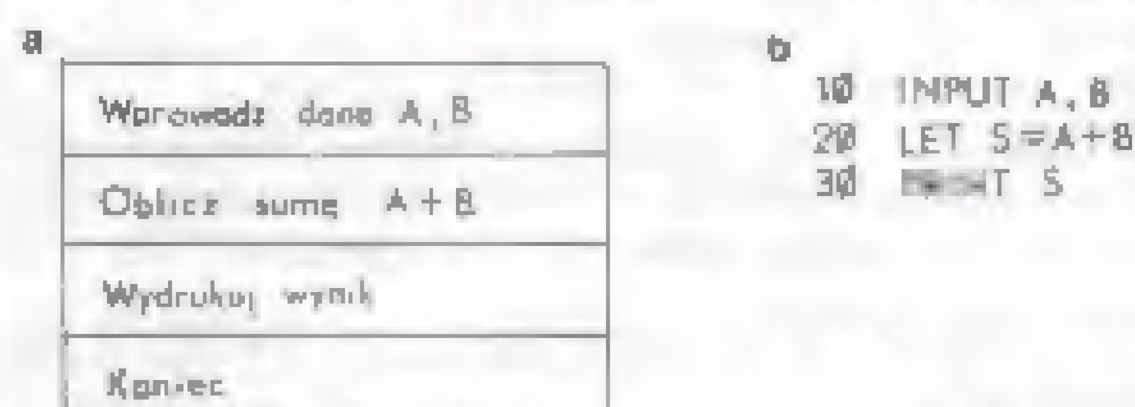
KOMPUTER W SZKOLE

GRAFICZNE PRZEDSTAWIENIE SCHEMATU DZIAŁANIA PROGRAMU — INACZEJ

Schemat działania programu, nazywany również schematem blokowym programu lub siecią działań, jest graficznym przedstawieniem kolejności wykonywania czynności wynikających z przyjętego algorytmu rozwiązania zadania. Schemat działania najczęściej tworzy się przy użyciu symboli graficznych połączonych strzałkami wskazującymi kolejność wykonywania czynności. Szerszy opis tego typu schematów blokowych można znaleźć w wielu pozycjach literatury.

Innym sposobem graficznego przedstawienia schematu działania programu są tzw. diagramy Nassi-Schneidermana, mniej znane w literaturze popularnonaukowej. Autor uważa, że warte są one przedstawienia szerszemu gronu Czytelników, gdyż lepiej spełniają swoje zadanie niż „stare" schematy blokowe. Diagramy Nassi-Schneidermana zmuszają niejako programistę do rozważnego stosowania instrukcji typu GOTO, jakże często nadużywanej przez początkujących użytkowników mikrokomputerów programujących w języku BASIC. A nikt chyba nie ma wątpliwości, że czytelność programów naszpikowanych instrukcjami GOTO jest zawsze co najmniej problematyczna. Poza tym diagramy wyrabiają u programisty dobry nawyk programowania strukturalnego.

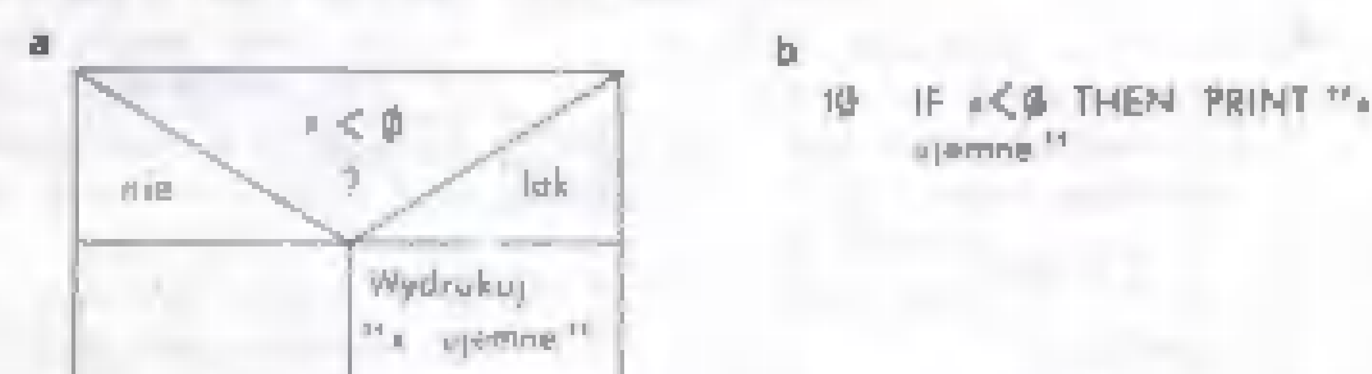
Najprostszym symbolem składowym diagramu jest prostokąt zawierający pojedynczą instrukcję BASIC-a lub grupę takich instrukcji. Można w nim umieszczać instrukcje typu LET, PRINT, INPUT, READ itp. lub słowny opis działania tych instrukcji (rys. 1).



Rys. 1. Diagram Nassi-Schneidermana (a) i odpowiadający mu fragment programu w Sinclair BASIC (komputer ZX Spectrum)

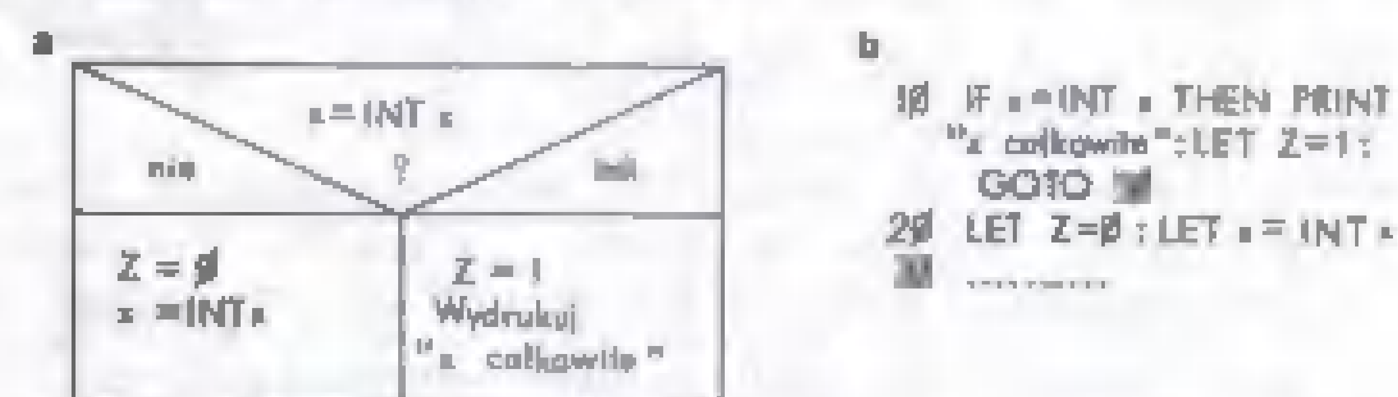
W tym prostym programie, składającym się wyłącznie z wykonywanych kolejno instrukcji BASIC-a, wpisanie działań w poszczególne prostokąty można przeprowadzić w dowolny sposób. W zasadzie wszystkie te instrukcje można było umieścić w jednym prostokącie. Wykonanie programu rozpoczyna się od czynności wymienionej w górnym prostokącie, „przechodzi" przez pozostałe prostokąty i kończy się na czynności wypisanej w dolnym prostokącie. Schemat całego programu lub jego logicznej części powinien znajdować się na jednej stronie. W odróżnieniu od schematu blokowego nie ma tutaj możliwości odwoływania się na inne strony, ponieważ takie działanie mogłoby znacznie zmniejszyć czytelność programu.

Instrukcji warunkowej typu IF ... THEN ... lub IF ... THEN ... ELSE ... odpowiadają symbole przedstawione na rysunkach 2a i 3a.



Rys. 2. Symbol odpowiadający instrukcji IF ... THEN ... (a) i jego odpowiednik w Sinclair BASIC (b)

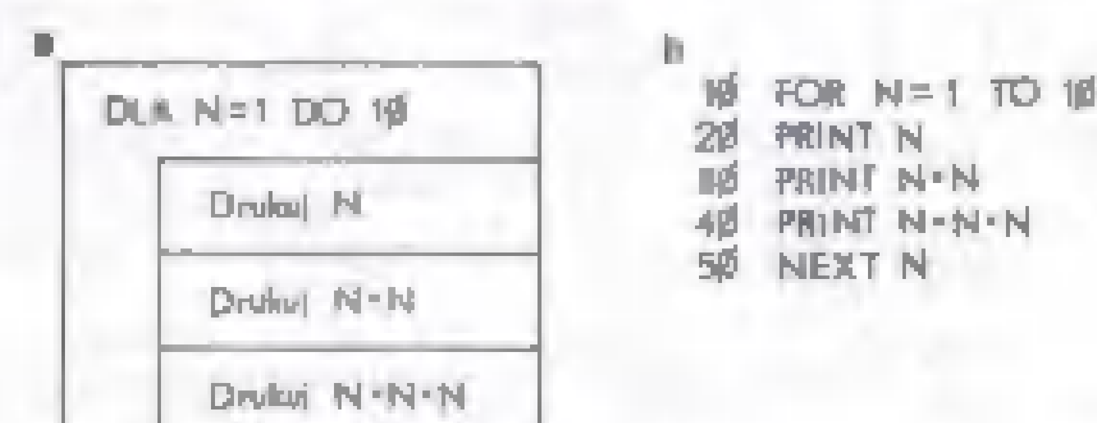
Jeżeli warunek x<Ø jest spełniony, to wykonywany jest fragment programu zawarty w prawym dolnym prostokącie, w przeciwnym przypadku — w lewym dolnym prostokącie. Jeśli wynikiem działania instrukcji IF jest tylko jedno działanie (instrukcja typu IF... THEN...), to jeden z niższych prostokątów należy zostawić pusty, tak jak na rys. 2a. W przypadku instrukcji typu IF... THEN... ELSE... wypełnione zostaną oba prostokąty (rys. 3a).



Rys. 3. Symbol odpowiadający instrukcji IF ... THEN ... ELSE ... (a) i jego odpowiednik w Sinclair BASIC

Należy zwrócić uwagę, że użycie instrukcji GOTO 3Ø zostało wykorzystane w celu realizacji instrukcji IF..THEN..ELSE nie występującej w dialekcie Sinclair BASIC (ZX Spectrum). Dla instrukcji GOTO nie przewidziano żadnego symbolu w diagramach Nassi-Schneidermana. Instrukcji tej należy używać łącznie z innymi instrukcjami w celu stworzenia standardowych struktur opisanych poniżej. Jest to właśnie jedna z zalet diagramów Nassi-Schneidermana.

Pokażemy teraz, w jaki sposób można w BASIC-u utworzyć różnego rodzaju pętle i jak przedstawić je na diagramach.

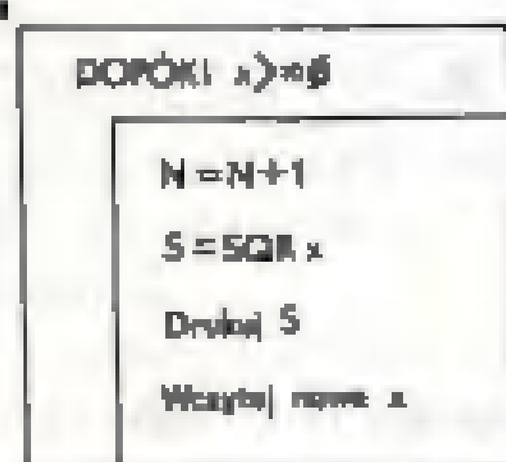


Rys. 4. Symbol odpowiadający instrukcji FOR ... NEXT (a) i jego odpowiednik w Sinclair BASIC (b)

Jeżeli z góry wiadomo ile razy dany fragment programu należy powtórzyć, to wykorzystujemy znaną pętlę FOR ... NEXT. Odpowiadający jej symbol przedstawiono na rys. 4a.

Innym rodzajem pętli jest pętla typu WHILE *(warunek)* DO *(instrukcja)*. Jest ona bardzo podobna do pętli FOR ... NEXT lecz nie jest z góry znana liczba powtórzeń. Na początku sprawdzany jest *warunek* umieszczony po słowie WHILE i jeżeli jest on spełniony, to wykonywana jest *instrukcja* (lub grupa instrukcji) znajdująca się po słowie DO. W przeciwnym przypadku program kończy pętlę i przechodzi do instrukcji znajdujących się bezpośrednio za pętlą. Symbol odpowiadający pętli WHILE ... DO przedstawiono na rys. 5a.

a



b

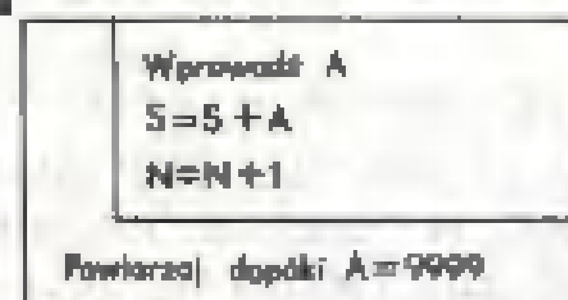
```
1Ø IF x<Ø THEN GOTO 7Ø
2Ø LET N=N+1
3Ø LET S=SQR x
4Ø PRINT S
5Ø INPUT x
6Ø GOTO 1Ø
7Ø .......
```

Rys. 5. Symbol odpowiadający pętli WHILE ... DO (a) i jego realizacja w Sinclair BASIC (b)

Zauważmy, że fragment programu z rys. 5b można zmodyfikować w następujący sposób 1ØIF NOT(x> =Ø) THEN GOTO 7Ø.

Następnym typem pętli jest pętla REPEAT *(instrukcja)* UNTIL *(warunek)*. W odróżnieniu od poprzednio omawianej pętli najpierw wykonywana jest *instrukcja* (lub grupa instrukcji) znajdująca się po słowie REPEAT, a następnie sprawdzany *warunek* umieszczony po słowie UNTIL. A więc fragment programu znajdujący się w pętli REPEAT ... UNTIL wykona się przynajmniej jeden raz. W przypadku pętli WHILE ... DO może zdarzyć się, że pętla nie zostanie wykonana, jeżeli warunek po słowie WHILE nie będzie spełniony już przed wykonaniem pętli. Symbol odpowiadający pętli REPEAT ... UNTIL przedstawiono na rys. 6a.

a



b

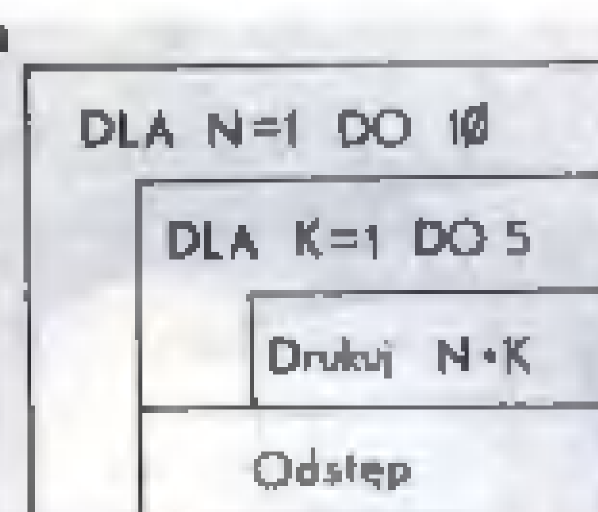
```
1Ø INPUT A
2Ø LET S=S+A
3Ø LET N=N+1
4Ø IF A<>9999 THEN GOTO 1Ø
```

Rys. 6. Symbol odpowiadający pętli REPEAT ... UNTIL (a) i jego realizacja w Sinclair BASIC (b)

Zbieżność symboli dla pętli FOR ... NEXT oraz WHILE ... DO nie jest przypadkowa. Większość wersji BASIC-a wykonuje pętle FOR ... NEXT podobnie jak pętle WHILE ... DO, tzn. najpierw sprawdzany jest warunek czy zmienna sterująca umieszczona po słowie FOR nie przekroczyła określonej wartości, a następnie wykonywany fragment programu zawarty w pętli. (Są jednak wersje BASIC-a, które „działają" na zasadzie pętli REPEAT ... UNTIL). Dla pętli zagnieżdżonych stosujemy symbol przedstawiony na rys. 7a.

W niektórych wersjach BASIC-a występuje instrukcja wyboru typu ON ... GOTO (BASIC ZX Spectrum nie posiada takiej instrukcji). Jest to „rozszerzenie" instrukcji IF ... THEN. Najpierw obliczane

a

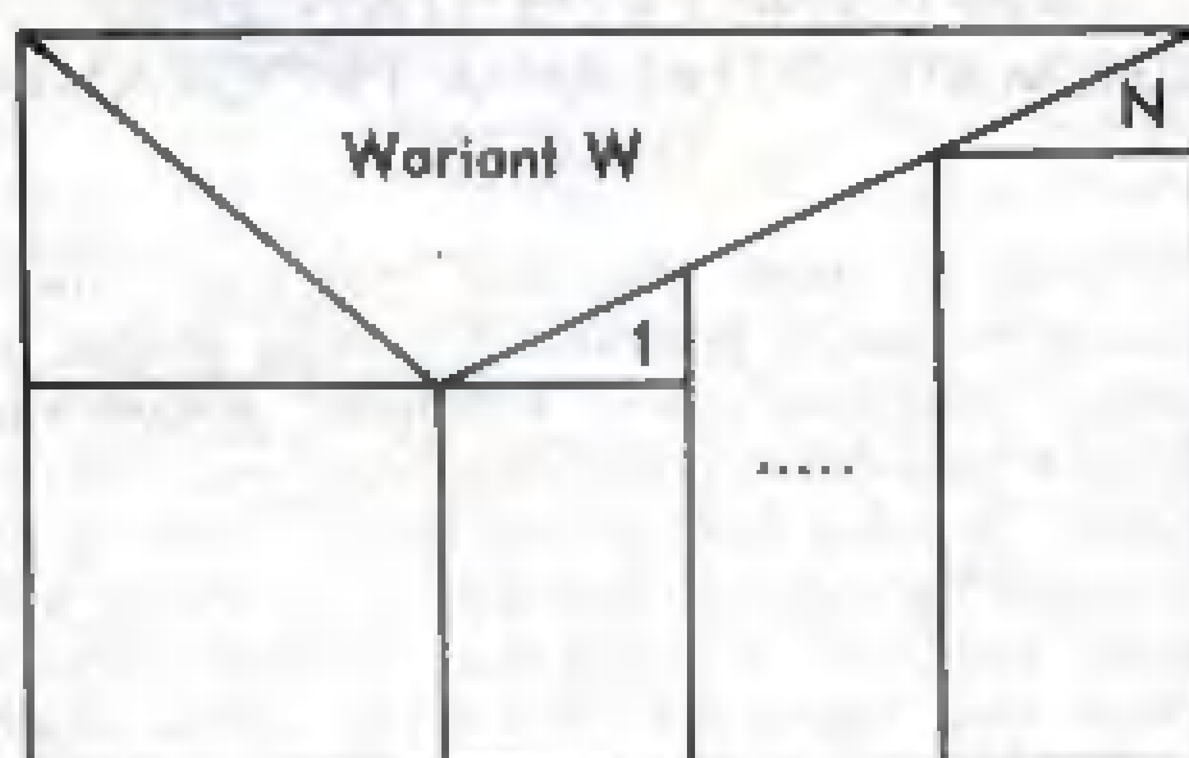


b

```
1Ø FOR N=1 TO 1Ø
2Ø FOR K=1 TO 5
3Ø PRINT N*K
4Ø NEXT K
5Ø PRINT
6Ø NEXT N
```

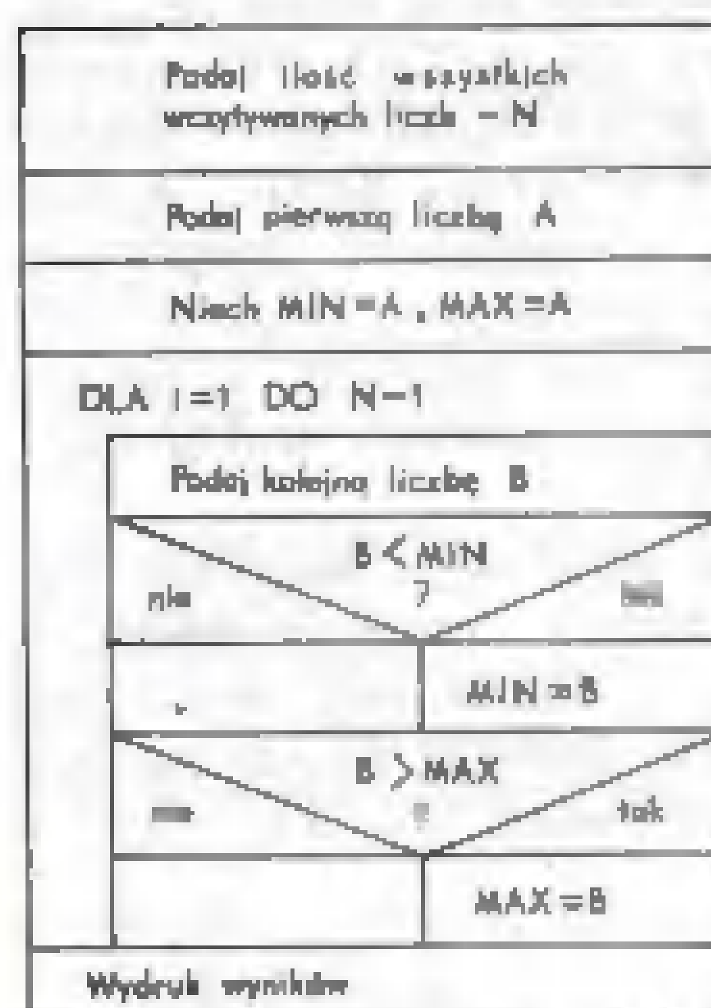
Rys. 7. Symbol pętli zagnieżdżonych (a) i przykład jego realizacji w Sinclair BASIC.

jest wyrażenie znajdujące się po słowie ON, a następnie wartość tego wyrażenia (zaokrągloną do liczby całkowitej) wykorzystuje się do wyboru jednego z numerów wierszy występujących po słowie GOTO. Jeśli wartość wyrażenia wynosi 1, to następuje skok do wiersza o numerze znajdującym się bezpośrednio po słowie GOTO itd. Symbol odpowiadający instrukcji ON ... GOTO ... przedstawiono na rys. 8.



Rys. 8. Symbol instrukcji wyboru typu ON ... GOTO ...

Na zakończenie przedstawimy przykładowy program w BASIC ZX Spectrum z wykorzystaniem diagramów Nassi-Schneidermana. Program wczytuje podawane przez użytkownika liczby, a następnie wyświetla największą i najmniejszą z nich (rys. 9).




```
1Ø INPUT "Podaj ilosc wszyst
   kich wczytywanych liczb"; N
2Ø INPUT "Podaj pierwsza licz
   be ";A
3Ø LET MIN=A : LET MAX=A
4Ø FOR I=1 TO N-1
5Ø INPUT "Podaj kolejna liczb
   e"; B
6Ø IF B<MIN THEN LET MIN=B
7Ø IF B>MAX THEN LET MAX=B
8Ø NEXT I
9Ø PRINT "MIN=" ; MIN
1ØØ PRINT "MAX=" ; MAX
```

Rys. 9. Przykładowy program w Sinclair BASIC (komputer ZX Spectrum). Opis programu w tekście

Wszystkim Czytelnikom zainteresowanym diagramami Nassi-Schneidermana, jak również wszystkim uczącym się programowania w języku BASIC polecam książkę w języku rosyjskim — „Programirowanie na Bejsikie", Moskwa 1987 (tłumaczenie z angielskiego — Brian C. Walsh — „Proper BASIC").

Zbigniew Krauze

KASOWALNE PAMIĘCI OPTYCZNE

Ogromna ilość przetwarzanych informacji jest charakterystyczną cechą współczesnej techniki komputerowej. Jak dotąd, bezkonkurencyjnym nośnikiem informacji okazały się być materiały magnetyczne. Materiały wykorzystywane do zapisu informacji cyfrowych (binarnych) charakteryzują się pętlą histerezy o dużej powierzchni (powierzchnia pętli histerezy decyduje o energii niezbędnej do przemagnesowania materiału) oraz dużą koercją (jeżeli zewnętrzne pole magnetyczne zostanie usunięte, magnetyzacja nie spada do zera, może ona zostać sprowadzona do zera, jeżeli zewnętrzne pole zostanie odwrócone i zwiększone do wartości zwanej koercją magnetyczną). Materiały magnetyczne, używane do zapisu informacji z dużymi gęstościami, powinny charakteryzować się szczególnie dużą powierzchnią pętli histerezy i wielkością koercji magnetycznej, tak aby można było łatwo odczytywać zapisaną informację oraz aby nie następował jej zanik.

Istnieje wiele metod odczytywania zapisanej na nośniku informacji. Najpopularniejsze z nich, stosowane we wszystkich popularnych magnetofonach, wykorzystują indukowanie się w przewodzie napięcia pod wpływem zmian strumienia magnetycznego, obejmowanego przez ten przewód. Dążenie do zmniejszenia szerokości ścieżek (a tym samym zwiększenia liczby cylindrów znajdujących się na dyskietce) powoduje zmniejszenie się wartości strumieni magnetycznych, przepływających przez magnetowód głowicy, a tym samym powoduje indukowanie się coraz mniejszych napięć w jej uzwojeniach.

Wzmacnianie małych napięć o częstotliwościach rzędu setek kHz, a czasem i kilku MHz napotyka w chwili obecnej na znaczne trudności wynikające z fizycznych uwarunkowań takich, jak np. szumy cieplne wzmacniacza, zakłócenia zewnętrzne itp. Można je ominąć stosując głowice magnetorezystancyjne, które zmieniają swoją rezystancję wraz ze zmianami zewnętrznego pola magnetycznego. Jednocześnie zmiany rezystancji tych głowic nie zależą od prędkości zmian zewnętrznego pola, a jedynie od wartości indukcji tego pola. Dodatkowo głowice te są bardzo cienkie (ok. 0,05 μm). Ich zalety są jednak okupione jedną wadą — mogą one jedynie odczytywać zapisaną informację, tak więc, aby stworzyć głowicę odczytująco-zapisującą, łączy się je z normalnymi głowicami, które pełnią rolę głowic zapisujących.

Okazuje się jednak, że zapisywać i odczytywać informacje na nośnikach magnetycznych można również zupełnie inaczej. Od dawna znany jest efekt Kerra, który objawia się skręceniem płaszczyzny polaryzacji światła odbitego od namagnesowanego pionowo nośnika magnetycznego. W zależności od tego, czy warstwa magnetyczna namagnesowana jest pionowo w dół, czy pionowo do góry, następuje skręcenie płaszczyzny polaryzacji odbijającego się światła przeciwnie lub zgodnie z kierunkiem ruchu wskazówek zegara. Stosując parę polaryzatorów można powstającą modulację płaszczyzny polaryzacji zmienić na modulację natężenia strumienia świetlnego i za pomocą fotodetektora (np. fotodioda PIN) przekształcić te zmiany na impulsy napięcia lub prądu. O ile sprawa odczytu jest bardzo prosta, o tyle zapis informacji jest już znacznie trudniejszy.

Dotychczas wydawało się, że możliwe jest tylko zapisanie informacji za pomocą światła, poprzez podniesienie temperatury nośnika magnetycznego. Specjalny nośnik napylony na dysk, wykonywany jest ze stopu metali ziem rzadkich. Stop ten charakteryzuje się dużą koercją magnetyczną w temperaturach pokojowych, malejącą szybko wraz ze wzrostem temperatury, co pozwala na łatwe przemagnesowanie go w podwyższonej temperaturze za pomocą niewielkich zewnętrznych pól magnetycznych. Podwyższenie temperatury jest dość łatwo uzyskać skupiając światło lasera na bardzo małej powierzchni ścieżki. Do odczytu można było wykorzystać ten sam laser, pracujący z o wiele mniejszą mocą. Wadą tego systemu była niemożność ponownego zapisu informacji na tej samej powierzchni. Ostatnio jednak udało się wymyślić nowe nośniki magnetyczne, będące stopami metali ziem rzadkich (gadolin i terb) z metalami takimi, jak żelazo i kobalt, mające bardzo obiecujące własności.

Momenty magnetyczne metali ziem rzadkich są skierowane przeciwnie do momentów magnetycznych metali przejściowych, co sprawia, że wypadkowy moment magnetyczny stopu jest różnicą pomiędzy dwoma momentami magnetycznymi składników stopu.

W niskich temperaturach dominuje magnetyzacja metali ziem rzadkich, w wyższych dominujące znaczenie ma magnetyzacja od metali przejściowych. Istnieją dwa punkty, w których wypadkowa magnetyzacja jest zerowa — pierwszy w temperaturze pośredniej, gdy wypadkowe magnetyzacje znoszą się, a drugi w wysokiej temperaturze, gdy chaotyczne ruchy atomów powodują równie chaotyczną orientację momentów magnetycznych poszczególnych atomów. Stop ten ma niezwykłą właściwość. Otóż po podgrzaniu części powierzchni pokrytej tym nośnikiem do wysokiej temperatury i ponownym schłodzeniu następuje przemagnesowanie podgrzanej powierzchni. Aby ponownie przemagnesować tę powierzchnię, wystarczy podgrzać jej centrum a zmiana rozprzestrzenia się aż do jej granic, pozostawiając nośnik w takim stanie namagnesowania, w jakim znajdował się uprzednio.

Zapisywanie tego typu nośnika, polega na jego uprzednim czytaniu i porównywaniu zapisanej na nim informacji z informacją, którą mamy zapisać. W momencie, gdy zapisana informacja różni się od nowej informacji, która ma zostać zapisana na powierzchni dysku, następuje krótki impuls z lasera zapisującego, powodujący przemagnesowanie „komórki". Obydwa lasery mogą być ogniskowane tym samym układem optycznym, wystarczy, by punkty na nośniku magnetycznym, w których zostają zogniskowane lasery, były przesunięte o kilka μm względem siebie.

System optycznego zapisu informacji, daje olbrzymie możliwości zagęszczania zapisu, gdyż szerokość ścieżki wynosi w tym systemie 0,1 μm, jednak znacznie istotniejsze znaczenie ma odporność systemu na drgania i przyspieszenia. Duża odporność na drgania jest wynikiem konieczności stosowania elektrodynamicznego systemu śledzenia ścieżki, prawie identycznego jak w odtwarzaczach laserowych Compact Disc. Wiąże się to z brakiem technicznych możliwości wyłącznie mechanicznego pozycjonowania głowicy z tak dużą dokładnością. Ponieważ głowica porusza się na wysokości rzędu 0,5 mm nad powierzchnią magnetyczną, nie zachodzi obawa dotknięcia przez głowicę nośnika, co często zdarza się w twardych dyskach, gdy na skutek drgań czy też napotkania drobiny kurzu, następuje zerwanie cienkiej poduszki powietrznej, na której unosi się głowica, powodując uszkodzenie dysku. Dokładność i pewność pozycjonowania głowicy jest jednak okupiona stosunkowo dużą masą systemu śledzącego (ok. 150 g), wielokrotnie większą od masy normalnych głowic magnetycznych. Stąd i duży czas odnajdywania cylindra, wynoszący ok. 0,1 s. Istnieją jednak możliwości poważnego ograniczenia masy głowicy optycznej poprzez zastosowanie do jej konstrukcji tzw. optyki zintegrowanej, wykonywanej technologią podobną niemal do technologii wykonywania układów scalonych.

Olbrzymie zalety nowego systemu gromadzenia informacji, przyczynią się z pewnością do jego rozpowszechnienia w przyszłości. Można nawet zaryzykować stwierdzenie, że będzie to najpopularniejszy typ pamięci na początku XXI w.

Piotr Postawka

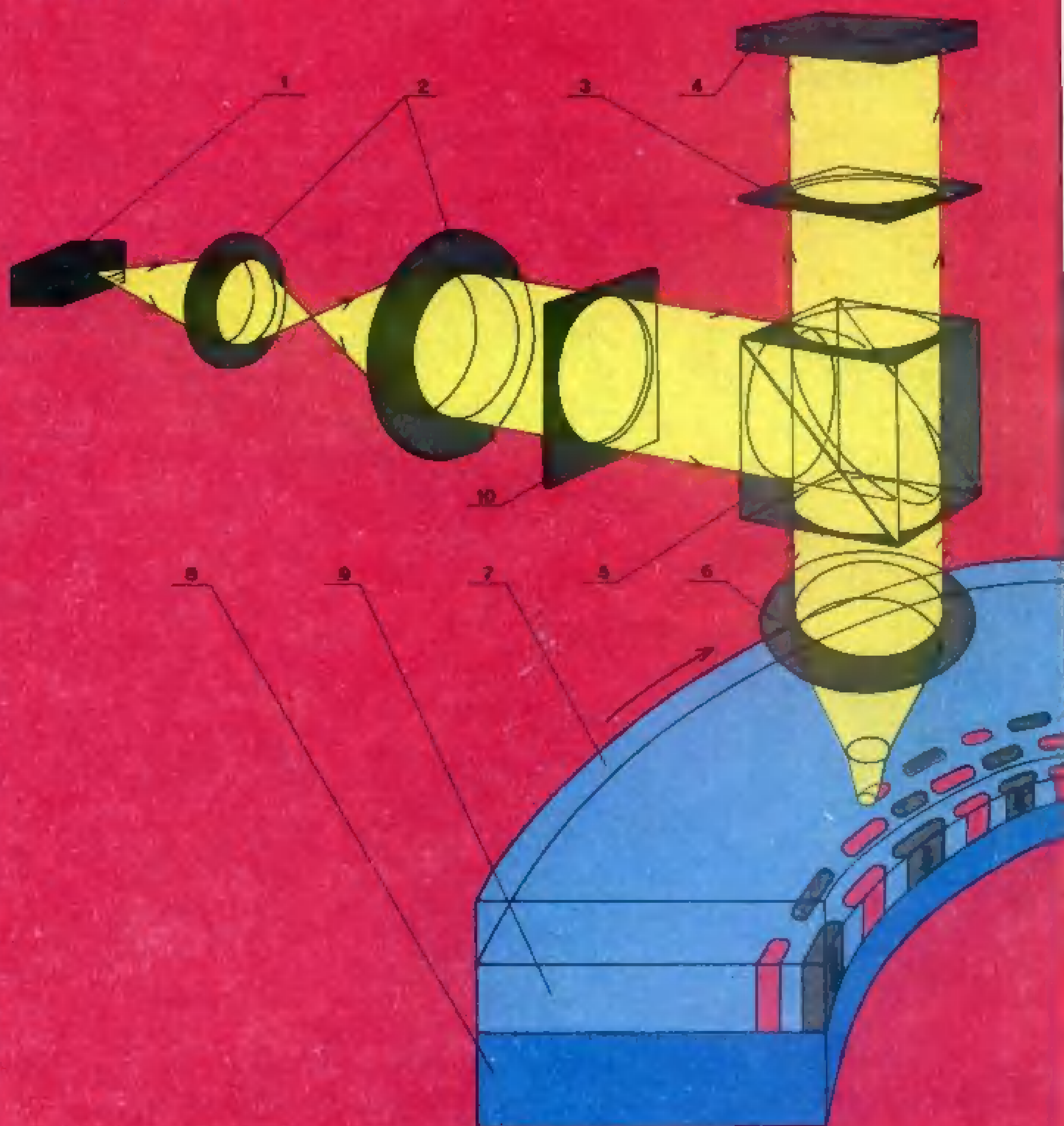
„Młody Technik — InforMik" wydaje Instytut Wydawniczy „Nasza Księgarnia"

Adres redakcji: ul. Spasowskiego 4, 00-389 Warszawa, lub skr. poczt. 380, 00-950 Warszawa. **Telefony:** centrala: 26-24-31 do 36. Dział Łączności z Czytelnikami — wewn. 60, pozostałe działy: wewn. 42 i 47. Redaktor naczelny: 26-26-27 lub wewn. 87.

Warunki prenumeraty: ogólnie obowiązujące w kraju. **W STAŁEJ SPRZEDAŻY INFORMIK JEST W SALONIE WYDAWNICZYM „NASZEJ KSIĘGARNI"** ul. Spasowskiego 4 A.

Redakcja zastrzega sobie prawo adiustacji i skracania nadesłanych materiałów. Artykułów nie zamówionych redakcja nie zwraca.

Druk: Zakłady Graficzne w Katowicach. Zam. 0623/1333/8
Nakład 60 000 + 315 egz.



Układ optyczny pamięci, umożliwiającej zapis i odczyt informacji na nośniku magnetycznym za pomocą światła lasera: **1** — fotodioda laserowa, **2** — układ soczewek kolimujących wiązkę, **3** — polaryzator analizujący, **4** — fotodetektor, **5** — układ rozdzielania wiązki, **6** — soczewka obiektywu, **7** — przezroczyste pokrycie, **8** — podłoże niemagnetyczne, **9** — warstwa magnetyczna, **10** — polaryzator